NAACL HLT 2009

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics

Proceedings of the Conference

May 31 – June 5, 2009
Boulder, Colorado

Production and Manufacturing by
Omnipress Inc.
2600 Anderson Street
Madison, WI 53707
USA

Sponsors:

- Rosetta Stone
- CNGL
- Microsoft Research
- Google
- AT&T
- Language Weaver
- J.D. Power
- IBM Research
- The Linguistic Data Consortium
- The Human Language Technology Center of Excellence at the Johns Hopkins University
- The Computational Language and Education Research Center at the University of Colorado at Boulder

©2009 The Association for Computational Linguistics

Order copies of this and other ACL proceedings from:

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

ISBN: 978-1-932432-41-1

Preface: General Chair

I am honored that the North American Chapter of the Association of Computational Linguistics (NAACL) has given me the opportunity, as General Conference Chair, to continue the NAACL HLT tradition of covering topics from all areas of Human Language Technology, which makes it possible for researchers to discuss algorithms and applications that cut across the fields of natural language processing (NLP), speech processing, and information retrieval (IR).

I have been very fortunate to work with a terrific group of Technical Program Co-Chairs: Michael Collins (NLP), Shri Narayanan (speech), Douglas W. Oard (IR), and Lucy Vanderwende (NLP). This year the technical program emphasizes the breadth and interdisciplinary nature of human language processing research. The plenary talks will stretch our thinking about how language is used by considering the application of language to vision in one case, and language as it relates to food in another. There are two special sessions with themes that cut across multiple sub-areas of HLT: Large Scale Language Processing and Speech Information Retrieval. We also recognize the increasing importance of industry in our field with a lunchtime panel discussion on the Next Big Applications in Industry, with thanks to Bill Dolan for organizing and moderating the discussion. Finally, we have a breadth of excellent technical papers in lecture and poster sessions, thanks to the efforts of our Senior Program Committee members, the many reviewers on the Program Committee who helped us keep to our schedule, and the Paper Awards Committee. Together they have done a great job in putting together an interesting technical program. It has also been a pleasure to work with Local Organizers Martha Palmer and Jim Martin, who have done a terrific job in hosting a meeting that shows us Colorado's character as well as offering a great technical program. I hope you enjoy your stay in beautiful Boulder, as you are learning about new ideas and networking with valued colleagues.

The tradition of NAACL HLT is that it incorporates many events, including tutorials and workshops that have expanded in scope such that they are almost as big as the main conference. As a result, many other people have played important roles in making the overall conference a success and representative of the breadth of HLT. Specifically, I thank Matthew Stone, Gokhan Tur and Diana Inkpen for their work as Publicity Chairs; Christy Doran and Eric Ringger for their work as Publications Chairs; Fred Popowich and Michael Johnston for serving as Demo Chairs; Tutorial Chairs Ciprian Chelba, Paul Kantor and Brian Roark for bringing us an outstanding slate of tutorials; Workshop Chairs Nizar Habash and Mark Hasegawa-Johnson for their efforts in choosing and supporting the 12 workshops that extend our program by two days; and the Student Co-Chairs of the Doctoral Consortium organizers Svetlana Stenchikova, Ulrich Germann and Chirag Shah working with faculty advisors Carolyn Rosé and Anoop Sarkar. Thanks also to Nicolas Nicolov for his efforts as NAACL HLT Sponsorship Chair, working in coordination with Sponsorship Chairs from other ACL regions. Of course, we greatly appreciate the support of our sponsors: Rosetta Stone, CNGL, Microsoft Research, Google, AT&T, Language Weaver, J.D. Power, IBM Research, the Linguistic Data Consortium, the Human Language Technology Center of Excellence at the Johns Hopkins University, and the Computational Language and Education Research Center at the University of Colorado at Boulder.

In organizing this conference, we have had a lot of support from the NAACL Board and the HLT Advisory Board. I would particularly like to thank Owen Rambow, Jennifer Chu-carroll, Chris Manning and Graeme Hirst for their help and advice. Last, but certainly not least, we are indebted to Priscilla Rasmussen for her expertise and support in running the conference.

Mari Ostendorf, University of Washington

Preface: Program Chairs

We welcome you to NAACL HLT 2009! The NAACL HLT program continues to include high-quality work in the areas of computational linguistics, information retrieval, and speech technology. This year, 260 full papers were submitted, of which 75 papers were accepted (giving a 29% acceptance rate); and 178 short papers were submitted, of which 71 were accepted (giving a 40% acceptance rate).

Two best paper awards were given at the conference, to "Unsupervised Morphological Segmentation with Log-Linear Models", by Hoifung Poon, Colin Cherry and Kristina Toutanova (this paper also received the best student paper award), and "11,001 New Features for Statistical Machine Translation", by David Chiang, Kevin Knight and Wei Wang. The senior program committee members for the conference nominated an initial set of papers that were candidates for the awards; the final decisions were then made by a committee chaired by Candace Sidner, and with Hal Daume III, Roland Kuhn, Ryan McDonald, and Mark Steedman as its other members. We would like to congratulate the authors, and thank the committee for their work in choosing these papers.

NAACL HLT 2009 consists of oral presentations of all full papers, oral or poster presentations of short papers, and tutorials and software demonstrations. We are delighted to have two keynote speakers: Antonio Torralba, with a talk "Understanding Visual Scenes", and Dan Jurafsky, with a talk "The Language of Food". In addition, we have a panel on emerging application areas in computational linguistics, chaired by Bill Dolan.

We would like to thank the authors for submitting a remarkable set of papers to the conference. The review process was organized through a two-tier system, with eighteen senior program committee (SPC) members, and 352 reviewers. The SPC members managed the review process for both the full and short paper submissions: each full paper received at least three reviews, and each short paper received at least two reviews. We are thoroughly indebted to the reviewers for all their work, and to the SPC members for the long hours they spent in evaluating the submissions. In addition, we would like to thank Rich Gerber and the START team for their help with the system that managed paper submissions and reviews; the local arrangement chairs, James Martin and Martha Palmer, for their help with organizing the program; and the publication chairs, Christy Doran and Eric Ringger, for putting together these proceedings. Finally, we are incredibly grateful to the general chair, Mari Ostendorf, for the invaluable advice and support that she provided throughout every step of the process.

We hope that you enjoy the conference!

Michael Collins, Massachusetts Institute of Technology
Shri Narayanan, University of Southern California
Douglas W. Oard, University of Maryland
Lucy Vanderwende, Microsoft Research

Organizers

General Chair:

Mari Ostendorf, University of Washington

Local Arrangements:

James Martin, University of Colorado
Martha Palmer, University of Colorado

Program Committee Chairs:

Michael Collins, Massachusetts Institute of Technology
Shri Narayanan, University of Southern California
Douglas W. Oard, University of Maryland
Lucy Vanderwende, Microsoft Research

Publicity Chairs:

Matthew Stone, Rutgers University
Gokhan Tur, SRI International
Diana Inkpen, University of Ottawa

Publications Chairs:

Christy Doran, MITRE
Eric Ringger, Brigham Young University

Tutorials Chairs:

Ciprian Chelba, Google
Paul Kantor, Rutgers University
Brian Roark, Oregon Health and Science University

Workshops Chairs:

Nizar Habash, Columbia University
Mark Hasegawa-Johnson, University of Illinois

Doctoral Consortium Organizers:

Carolyn Rosé, Faculty Chair, CMU
Anoop Sarkar, Faculty Chair, Simon Fraser University

Svetlana Stoyachev, Student Co-Chair, Stony Brook University
Ulrich Germann, Student Co-Chair, University of Toronto
Chirag Shah, Student Co-Chair, University of North Carolina

Demo Chairs:

Fred Popowich, Simon Fraser University
Michael Johnston, AT&T

Sponsorship Committee:

Nicolas Nicolov (Local Chair)
Hitoshi Isahara and Kim-Teng Lua (Asian ACL Rrepresentatives)
Philipp Koehn and Josef van Genabith (European ACL Representatives)
Srinivas Bangalore and Christy Doran (American ACL Representatives)

Program Committee

Senior Program Committee Members:

Michiel Bacchiani, Google
Regina Barzilay, Massachusetts Institute of Technology
Kenneth W. Church, Microsoft Research
Charles L. A. Clarke, University of Waterloo
Eric Fosler-Lussier, Ohio State University
Sharon Goldwater, University of Edinburgh
Julia Hirschberg, Columbia University
Jimmy Huang, York University
Mark Johnson, Brown University
Philipp Koehn, University of Edinburgh
Roland Kuhn, National Research Council of Canada, IIT
Gina-Anne Levow, University of Manchester
Dekang Lin, Google
Ryan McDonald, Google
Premkumar Natarajan, BBN Technologies
Patrick Pantel, Yahoo! Labs
Kristina Toutanova, Microsoft Research
Geoff Zweig, Microsoft Research

Paper Award Committee:

Candace Sidner, Chair, BAE Systems AIT
Hal Daumé III, University of Utah
Roland Kuhn, NRC Institute for Information Technology
Ryan McDonald, Google Inc.
Mark Steedman, University of Edinburgh

Program Committee Members:

Stephen Abney
Meni Adler
Eugene Agichtein
Eneko Agirre
Lars Ahrenberg
Adam Albright
Enrique Alfonseca
Afra Alishahi
Sophia Ananiadou
Shankar Ananthakrishnan
Bill Andreopoulos
Galen Andrew
Walter Andrews
Masayuki Asahara
Necip Fazil Ayan
Mark Baillie
Timothy Baldwin
Roberto Basili
Ron Bekkerman
Sabine Bergler
Shane Bergsma
Rahul Bhagat
Dan Bikel
Mikhail Bilenko

Alexandra Birch
Alan Black
Sasha Blair-Goldensohn
John Blitzer
Paul Boersma
Johan Bos
Alexandre Bouchard-Côté
S.R.K. Branavan
Chris Brew
Ted Briscoe
Chris Brockett
Stefan Buettcher
Razvan Bunescu
Jill Burstein
Cory Butz
William Byrne
Chris Callison-Burch
Claire Cardie
Giuseppe Carenini
Marine Carpuat
Xavier Carreras
Francisco Casacuberta
Joyce Chai
Yllias Chali
Nate Chambers
Jason Chang
Eugene Charniak
Ciprian Chelba
Harr Chen
Colin Cherry
David Chiang
Tat-Seng Chua
Grace Chung
Massimiliano Ciaramita
Stephen Clark
Peter Clark
Mark Craven
Mathias Creutz
Aron Culotta
James Cussens
Robert Dale
Cristian Danescu Niculescu-Mizil
Hal Daumé III
Guy De Pauw
John DeNero
Barbara Di Eugenio
Mona Diab
Bill Dolan
Christy Doran
Doug Downey
Mark Dredze
Markus Dreyer
Rebecca Dridan
Kevin Duh
Chris Dyer
Andreas Eisele
Jacob Eisenstein
Jason Eisner
Michael Elhadad
Noemie Elhadad
Mark Ellison
Micha Elsner
Dominique Estival
Oren Etzioni
Hui Fang
Marcello Federico
Paolo Ferragina
Jenny Finkel
Erin Fitzgerald
Radu Florian
George Foster
Dayne Freitag
Pascale Fung
Robert Gaizauskas
Michael Gamon
Kuzman Ganchev
Jianfeng Gao
Claire Gardent
Stuart Geman
Ulrich Germann
Shlomo Geva
Mazin Gilbert
Daniel Gildea
Jesus Gimenez
Roxana Girju
Randy Goebel
John Goldsmith
Ralph Grishman
Asela Gunawardana
Gholamreza Haffari
Aria Haghighi
Udo Hahn

Dilek Hakkani-Tür
Keith Hall
Hyoil Han
Mary Harper
Saša Hasan
Mark Hasegawa-Johnson
Timothy J. Hazen
Xiaodong He
William Headden
Peter Heeman
James Henderson
Iris Hendrickx
Graeme Hirst
Hieu Hoang
Kristy Hollingshead
Mark Hopkins
Vronique Hoste
Chu-Ren Huang
Liang Huang
Rebecca Hwa
Diana Inkpen
Abe Ittycheriah
Gaja Jarosz
Heng Ji
Richard Johansson
Howard Johnson
Rie Johnson
Doug Jones
Gareth Jones
Aravind Joshi
Min-Yen Kan
Chia-lin Kao
Nikiforos Karamanis
Rohit Kate
Vlado Keselj
Shahram Khadivi
Sanjeev Khudanpur
Adam Kilgarriff
Jin-Dong Kim
Owen Kimball
Dan Klein
Kevin Knight
Mamoru Komachi
Grzegorz Kondrak
Terry Koo
Anna Korhonen
Kimmo Koskenniemi
Emiel Krahmer
Jonas Kuhn
Shankar Kumar
Christian König
Philippe Langlais
Mirella Lapata
Alex Lascarides
Alon Lavie
Claudia Leacock
Lillian Lee
Yoong Keok Lee
James Lester
Gregor Leusch
Roger Levy
David Lewis
Wei Li
Xiao Li
Haizhou Li
Hang Li
Ping Li
Percy Liang
Hank Liao
Jimmy Lin
Chin-Yew Lin
Bing Liu
Yang Liu
Tie-Yan Liu
Andrej Ljolje
Adam Lopez
Alex Lopez-Ortiz
Bill MacCartney
Nitin Madnani
Bernardo Magnini
Jonathan Mamou
Suresh Manandhar
Lidia Mangu
Gideon Mann
Chris Manning
Daniel Marcu
Evgeny Matusov
Arne Mauser
David McAllester
Andrew McCallum
Diana McCarthy
David McClosky

Kathy McCoy
Kathleen McKeown
Susan McRoy
Qiaozhu Mei
Paola Merlo
Rada Mihalcea
Yusuke Miyao
Saif Mohammad
Dan Moldovan
Bob Moore
Richard Moot
Pedro Moreno
Dragos Munteanu
Smaranda Muresan
Muthu Muthukrishnan
Tetsuji Nakagawa
Preslav Nakov
Ani Nenkova
Hermann Ney
Hwee Tou Ng
Vincent Ng
Raymond Ng
Patrick Nguyen
Jian-Yun Nie
Joakim Nivre
Franz Och
Kemal Oflazer
Scott Olsson
Luca Onnis
Miles Osborne
Tim Paek
Bo Pang
Marius Pasca
Rebecca Passonneau
Matthias Paulik
Ted Pedersen
Marco Pennacchiotti
Mati Pentus
Amy Perfors
Slav Petrov
Joseph Picone
Janet Pierrehumbert
Livia Polanyi
Hoifung Poon
Ana-Maria Popescu
Maja Popovic
Fred Popowich
John Prager
Rohit Prasad
Partha Pratim Talukdar
Matthew Purver
Chris Quirk
Drago Radev
Rajat Raina
Daniel Ramage
Owen Rambow
Vivek Kumar Rangarajan Sridhar
Deepak Ravichandran
Stefan Riezler
Ellen Riloff
Eric Ringger
Brian Roark
Barbara Rosario
Dan Roth
Alex Rudnicky
Marta Ruiz
Anton Rytting
Kenji Sagae
Johan Schalkwyk
David Schlangen
Tanja Schultz
Petr Schwarz
Holger Schwenk
Satoshi Sekine
Mike Seltzer
Stephanie Seneff
Wade Shen
Stuart Shieber
Luo Si
Michel Simard
Olivier Siohan
Kevin Small
David Smith
Noah Smith
Mark Smucker
Rion Snow
Ben Snyder
Radu Soricut
Richard Sproat
Amit Srivastava
David Stallard
Mark Steedman

Mark Stevenson
Michael Strube
Amarnag Subramanya
Torsten Suel
Eiichiro Sumita
Charles Sutton
David Talbot
Ben Taskar
Yee Whye Teh
Simone Teufel
Joerg Tiedemann
Christoph Tillmann
Ivan Titov
Isabel Trancoso
David Traum
Andrew Trotman
Peter Turney
Nicola Ueffing
Jay Urbain
Antal van den Bosch
Benjamin van Durme
Olga Vechtomova
Dimitra Vergyri
Evelyne Viegas
David Vilar
Ye-Yi Wang
Qin Wang
Nigel Ward
Taro Watanabe
Bonnie Webber
MIchael White
Richard Wicentowski
Jason Williams
Shuly Wintner
Dekai Wu
Mingfang Wu
Peng Xu
Roman Yangarber
Alex Yates
Zheng Ye
Scott Wen-tau Yih
Chen Yu
Dong Yu
Fabio Massimo Zanzotto
Richard Zens
Luke Zettlemoyer
Hao Zhang
Ming Zhou
Wei Zhou
Bowen Zhou
Jerry Zhu
Jianhan Zhu
Andreas Zollmann

Table of Contents

Conference Program Overview

Monday, June 1, 2009

9:00–10:10 Plenary Session – Invited Talk by Antonio Torralba: *Understanding Visual Scenes*

10:40–11:20 Session 1A: Semantics
Session 1B: Multilingual Processing / Morphology and Phonology
Session 1C: Syntax and Parsing
Student Research Workshop Session 1

2:00–3:30 Short Paper Presentations:
Session 2A: Machine Translation
Session 2B: Information Retrieval / Information Extraction / Sentiment
Session 2C: Dialog / Speech / Semantics
Student Research Workshop Session 2

4:00–5:40 Session 3A: Machine Translation
Session 3B: Semantics
Session 3C: Information Retrieval
Student Research Workshop Session 3

6:30–9:30 Poster and Demo Session
Student Research Workshop Poster Session

Tuesday, June 2, 2009

9:00-10:10 Plenary Session: Paper Award Presentations

10:10–11:40 Session 4A: Machine Translation
Session 4B: Sentiment Analysis / Information Extraction
Session 4C: Machine Learning / Morphology and Phonology

2:00–3:30 Short Paper Presentations:
Session 5A: Machine Translation / Generation / Semantics
Session 5B: Machine Learning / Syntax
Session 5C: SPECIAL SESSION – Speech Indexing and Retrieval

4:00–5:15 Session 6A: Syntax and Parsing
Session 6B: Discourse and Summarization
Session 6C: Spoken Language Systems

Wednesday, June 3, 2009

9:00–10:10	Plenary Session – Invited Talk by Dan Jurafsky: *Ketchup, Espresso, and Chocolate Chip Cookies: Travels in the Language of Food*
10:40–12:20	Session 7A: Machine Translation Session 7B: Speech Recognition and Language Modeling Session 7C: Sentiment Analysis
12:40-1:40	Panel Discussion: *Emerging Application Areas in Computational Linguistics*
1:40–2:30	NAACL Business Meeting
2:30–3:45	Session 8A: Large-scale NLP Session 8B: Syntax and Parsing Session 8C: Discourse and Summarization
4:15–5:30	Session 9A: Machine Learning Session 9B: Dialog Systems Session 9C: Syntax and Parsing

Conference Program

Monday, June 1, 2009

Plenary Session

9:00–10:10 Welcome and Invited Talk: *Understanding Visual Scenes*
Antonio Torralba

10:10–10:40 **Break**

Session 1A: Semantics

10:40–11:05 *Subjectivity Recognition on Word Senses via Semi-supervised Mincuts*
Fangzhong Su and Katja Markert

11:05–11:30 *Integrating Knowledge for Subjectivity Sense Labeling*
Yaw Gyamfi, Janyce Wiebe, Rada Mihalcea and Cem Akkaya

11:30–11:55 *A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches*
Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca and Aitor Soroa

11:55–12:20 *A Fully Unsupervised Word Sense Disambiguation Method Using Dependency Knowledge*
Ping Chen, Wei Ding, Chris Bowes and David Brown

Session 1B: Multilingual Processing / Morphology and Phonology

10:40–11:05 *Learning Phoneme Mappings for Transliteration without Parallel Data*
Sujith Ravi and Kevin Knight

11:05–11:30 *A Corpus-Based Approach for the Prediction of Language Impairment in Monolingual English and Spanish-English Bilingual Children*
Keyur Gabani, Melissa Sherman, Thamar Solorio, Yang Liu, Lisa Bedore and Elizabeth Peña

11:30–11:55 *A Discriminative Latent Variable Chinese Segmenter with Hybrid Word/Character Information*
Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka and Jun'ichi Tsujii

11:55–12:20 *Improved Reconstruction of Protolanguage Word Forms*
Alexandre Bouchard-Côté, Thomas L. Griffiths and Dan Klein

Monday, June 1, 2009 (continued)

Session 1C: Syntax and Parsing

10:40–11:05 *Shared Logistic Normal Distributions for Soft Parameter Tying in Unsupervised Grammar Induction*
Shay Cohen and Noah A. Smith

11:05–11:30 *Adding More Languages Improves Unsupervised Multilingual Part-of-Speech Tagging: a Bayesian Non-Parametric Approach*
Benjamin Snyder, Tahira Naseem, Jacob Eisenstein and Regina Barzilay

11:30–11:55 *Efficiently Parsable Extensions to Tree-Local Multicomponent TAG*
Rebecca Nesson and Stuart Shieber

11:55–12:20 *Improving Unsupervised Dependency Parsing with Richer Contexts and Smoothing*
William P. Headden III, Mark Johnson and David McClosky

Student Research Workshop Session 1:

Note: all student research workshop papers are located in the Companion volume of the proceedings

10:40–11:10 *Classifier Combination Techniques Applied to Coreference Resolution*
Smita Vemulapalli, Xiaoqiang Luo, John F. Pitrelli and Imed Zitouni

11:15–11:45 *Solving the "Who's Mark Johnson Puzzle": Information Extraction Based Cross Document Coreference*
Jian Huang, Sarah M. Taylor, Jonathan L. Smith, Konstantinos A. Fotiadis and C. Lee Giles

11:50–12:20 *Exploring Topic Continuation Follow-up Questions using Machine Learning*
Manuel Kirschner and Raffaella Bernardi

12:20–2:00 **Lunch Break**

Monday, June 1, 2009 (continued)

Session 2A: Short Paper Presentations: Machine Translation

Note: all short papers are located in the Companion volume of the proceedings

2:00–2:15 *Cohesive Constraints in A Beam Search Phrase-based Decoder*
Nguyen Bach, Stephan Vogel and Colin Cherry

2:15–2:30 *Revisiting Optimal Decoding for IBM Machine Translation Model 4*
James Clarke and Sebastian Riedel

2:30–2:45 *Efficient Extraction of Oracle-best Translations from Hypergraphs*
Zhifei Li and Sanjeev Khudanpur

2:45–3:00 *Semantic Roles for SMT: A Hybrid Two-Pass Model*
Dekai Wu and Pascale Fung

3:00–3:15 *Comparison of Extended Lexicon Models in Search and Rescoring for SMT*
Saša Hasan and Hermann Ney

3:15–3:30 *Simplex Armijo Downhill Algorithm for Optimizing Statistical Machine Translation System Parameters*
Bing Zhao and Shengyuan Chen

Session 2B: Short Paper Presentations: Information Retrieval / Information Extraction / Sentiment

Note: all short papers are located in the Companion volume of the proceedings

2:00–2:15 *Translation Corpus Source and Size in Bilingual Retrieval*
Paul McNamee, James Mayfield and Charles Nicholas

2:15–2:30 *Large-scale Computation of Distributional Similarities for Queries*
Enrique Alfonseca, Keith Hall and Silvana Hartmann

2:30–2:45 *Text Categorization from Category Name via Lexical Reference*
Libby Barak, Ido Dagan and Eyal Shnarch

2:45–3:00 *Identifying Types of Claims in Online Customer Reviews*
Shilpa Arora, Mahesh Joshi and Carolyn Rose

Monday, June 1, 2009 (continued)

3:00–3:15 *Towards Automatic Image Region Annotation - Image Region Textual Coreference Resolution*
Emilia Apostolova and Dina Demner-Fushman

3:15–3:30 *TESLA: A Tool for Annotating Geospatial Language Corpora*
Nate Blaylock, Bradley Swain and James Allen

Session 2C: Short Paper Presentations: Dialog / Speech / Semantics

Note: all short papers are located in the Companion volume of the proceedings

2:00–2:15 *Modeling Dialogue Structure with Adjacency Pair Analysis and Hidden Markov Models*
Kristy Elizabeth Boyer, Robert Phillips, Eun Young Ha, Michael Wallis, Mladen Vouk and James Lester

2:15–2:30 *Towards Natural Language Understanding of Partial Speech Recognition Results in Dialogue Systems*
Kenji Sagae, Gwen Christian, David DeVault and David Traum

2:30–2:45 *Spherical Discriminant Analysis in Semi-supervised Speaker Clustering*
Hao Tang, Stephen Chu and Thomas Huang

2:45–3:00 *Learning Bayesian Networks for Semantic Frame Composition in a Spoken Dialog System*
Marie-Jean Meurs, Fabrice Lefvre and Renato De Mori

3:00–3:15 *Evaluation of a System for Noun Concepts Acquisition from Utterances about Images (SINCA) Using Daily Conversation Data*
Yuzu Uchida and Kenji Araki

3:15–3:30 *Web and Corpus Methods for Malay Count Classifier Prediction*
Jeremy Nicholson and Timothy Baldwin

Monday, June 1, 2009 (continued)

Student Research Workshop Session 2

Note: all student research workshop papers are located in the Companion volume of the proceedings

2:00–2:30 *Sentence Realisation from Bag of Words with Dependency Constraints*
Karthik Gali and Sriram Venkatapathy

2:35–3:05 *Using Language Modeling to Select Useful Annotation Data*
Dmitriy Dligach and Martha Palmer

3:30–4:00 **Break**

Session 3A: Machine Translation

4:00–4:25 *Context-Dependent Alignment Models for Statistical Machine Translation*
Jamie Brunning, Adrià de Gispert and William Byrne

4:25–4:50 *Graph-based Learning for Statistical Machine Translation*
Andrei Alexandrescu and Katrin Kirchhoff

4:50–5:15 *Intersecting Multilingual Data for Faster and Better Statistical Translations*
Yu Chen, Martin Kay and Andreas Eisele

5:15–5:40 No Presentation

Session 3B: Semantics

4:00–4:25 *Without a 'doubt'? Unsupervised Discovery of Downward-Entailing Operators*
Cristian Danescu-Niculescu-Mizil, Lillian Lee and Richard Ducott

4:25–4:50 *The Role of Implicit Argumentation in Nominal SRL*
Matthew Gerber, Joyce Chai and Adam Meyers

4:50–5:15 *Jointly Identifying Predicates, Arguments and Senses using Markov Logic*
Ivan Meza-Ruiz and Sebastian Riedel

Monday, June 1, 2009 (continued)

5:15–5:40 *Structured Generative Models for Unsupervised Named-Entity Clustering*
Micha Elsner, Eugene Charniak and Mark Johnson

Session 3C: Information Retrieval

4:00–4:25 *Hierarchical Dirichlet Trees for Information Retrieval*
Gholamreza Haffari and Yee Whye Teh

4:25–4:50 *Phrase-Based Query Degradation Modeling for Vocabulary-Independent Ranked Utterance Retrieval*
J. Scott Olsson and Douglas W. Oard

4:50–5:15 *Japanese Query Alteration Based on Lexical Semantic Similarity*
Masato Hagiwara and Hisami Suzuki

5:15–5:40 *Context-based Message Expansion for Disentanglement of Interleaved Text Conversations*
Lidan Wang and Douglas W. Oard

Student Research Workshop Session 3

Note: all student research workshop papers are located in the Companion volume of the proceedings

4:00–4:30 *Pronunciation Modeling in Spelling Correction for Writers of English as a Foreign Language*
Adriane Boyd

4:35–5:05 *Building a Semantic Lexicon of English Nouns via Bootstrapping*
Ting Qian, Benjamin Van Durme and Lenhart Schubert

5:10–5:40 *Multiple Word Alignment with Profile Hidden Markov Models*
Aditya Bhargava and Grzegorz Kondrak

6:30–9:30 **Poster and Demo Session**

Note: all short papers and demo abstracts are located in the Companion volume of the proceedings

Minimum Bayes Risk Combination of Translation Hypotheses from Alternative Morphological Decompositions
Adri de Gispert, Sami Virpioja, Mikko Kurimo and William Byrne

Monday, June 1, 2009 (continued)

Generating Synthetic Children's Acoustic Models from Adult Models
Andreas Hagen, Bryan Pellom and Kadri Hacioglu

Detecting Pitch Accents at the Word, Syllable and Vowel Level
Andrew Rosenberg and Julia Hirschberg

Shallow Semantic Parsing for Spoken Language Understanding
Bonaventura Coppola, Alessandro Moschitti and Giuseppe Riccardi

Automatic Agenda Graph Construction from Human-Human Dialogs using Clustering Method
Cheongjae Lee, Sangkeun Jung, Kyungduk Kim and Gary Geunbae Lee

A Simple Sentence-Level Extraction Algorithm for Comparable Data
Christoph Tillmann and Jian-ming Xu

Learning Combination Features with L1 Regularization
Daisuke Okanohara and Jun'ichi Tsujii

Multi-scale Personalization for Voice Search
Daniel Bolanos, Geoffrey Zweig and Patrick Nguyen

The Importance of Sub-Utterance Prosody in Predicting Level of Certainty
Heather Pon-Barry and Stuart Shieber

Using Integer Linear Programming for Detecting Speech Disfluencies
Kallirroi Georgila

Contrastive Summarization: An Experiment with Consumer Reviews
Kevin Lerman and Ryan McDonald

Topic Identification Using Wikipedia Graph Centrality
Kino Coursey and Rada Mihalcea

Extracting Bilingual Dictionary from Comparable Corpora with Dependency Heterogeneity
Kun Yu and Junichi Tsujii

Domain Adaptation with Artificial Data for Semantic Parsing of Speech
Lonneke van der Plas, James Henderson and Paola Merlo

Extending Pronunciation Lexicons via Non-phonemic Respellings
Lucian Galescu

A Speech Understanding Framework that Uses Multiple Language Models and Multiple Understanding Models
Masaki Katsumaru, Mikio Nakano, Kazunori Komatani, Kotaro Funakoshi, Tetsuya Ogata and Hiroshi G. Okuno

Taking into Account the Differences between Actively and Passively Acquired Data: The Case of Active Learning with Support Vector Machines for Imbalanced Datasets
Michael Bloodgood and Vijay Shanker

Faster MT Decoding Through Pervasive Laziness
Michael Pust and Kevin Knight

Evaluating the Syntactic Transformations in Gold Standard Corpora for Statistical Sentence Compression
Naman K Gupta, Sourish Chaudhuri and Carolyn P Rose

Incremental Adaptation of Speech-to-Speech Translation
Nguyen Bach, Roger Hsiao, Matthias Eck, Paisarn Charoenpornsawat, Stephan Vogel, Tanja Schultz, Ian Lane, Alex Waibel and Alan Black

Name Perplexity
Octavian Popescu

Answer Credibility: A Language Modeling Approach to Answer Validation
Protima Banerjee and Hyoil Han

Exploiting Named Entity Classes in CCG Surface Realization
Rajakrishnan Rajkumar, Michael White and Dominic Espinosa

Search Engine Adaptation by Feedback Control Adjustment for Time-sensitive Query
Ruiqiang zhang, yi Chang, Zhaohui Zheng, Donald Metzler and Jian-yun Nie

A Local Tree Alignment-based Soft Pattern Matching Approach for Information Extraction
Seokhwan Kim, Minwoo Jeong and Gary Geunbae Lee

Classifying Factored Genres with Part-of-Speech Histograms
Sergey Feldman, Marius Marin, Julie Medero and Mari Ostendorf

Towards Effective Sentence Simplification for Automatic Processing of Biomedical Text
Siddhartha Jonnalagadda, Luis Tari, Jrg Hakenberg, Chitta Baral and Graciela Gonzalez

Improving SCL Model for Sentiment-Transfer Learning
Songbo Tan and Xueqi Cheng

MICA: A Probabilistic Dependency Parser Based on Tree Insertion Grammars (Application Note)
Srinivas Bangalore, Pierre Boullier, Alexis Nasr, Owen Rambow and Benot Sagot

Lexical and Syntactic Adaptation and Their Impact in Deployed Spoken Dialog Systems
Svetlana Stoyanchev and Amanda Stent

Analysing Recognition Errors in Unlimited-Vocabulary Speech Recognition
Teemu Hirsimäki and Mikko Kurimo

The independence of dimensions in multidimensional dialogue act annotation
Volha Petukhova and Harry Bunt

Improving Coreference Resolution by Using Conversational Metadata
Xiaoqiang Luo, Radu Florian and Todd Ward

Using N-gram based Features for Machine Translation System Combination
Yong Zhao and Xiaodong He

Language Specific Issue and Feature Exploration in Chinese Event Extraction
Zheng Chen and Heng Ji

Improving A Simple Bigram HMM Part-of-Speech Tagger by Latent Annotation and Self-Training
Zhongqiang Huang, Vladimir Eidelman and Mary Harper

6:30–9:30 **Student Research Workshop Poster Session**

Note: all student research workshop papers are located in the Companion volume of the proceedings

Also: All papers presented in the morning and afternoon sessions of the student research workshop will also be shown as posters.

Using Emotion to Gain Rapport in a Spoken Dialog System
Jaime Acosta

Interactive Annotation Learning with Indirect Feature Voting
Shilpa Arora and Eric Nyberg

Monday, June 1, 2009 (continued)

Loss-Sensitive Discriminative Training of Machine Transliteration Models
Kedar Bellare, Koby Crammer and Dayne Freitag

Syntactic Tree-based Relation Extraction Using a Generalization of Collins and Duffy Convolution Tree Kernel
Mahdy Khayyamian, Seyed Abolghasem Mirroshandel and Hassan Abolhassani

Towards Building a Competitive Opinion Summarization System: Challenges and Keys
Elena Lloret, Alexandra Balahur, Manuel Palomar and Andres Montoyo

Domain-Independent Shallow Sentence Ordering
Thade Nahnsen

Towards Unsupervised Recognition of Dialogue Acts
Nicole Novielli and Carlo Strapparava

Modeling Letter-to-Phoneme Conversion as a Phrase Based Statistical Machine Translation Problem with Minimum Error Rate Training
Taraka Rama, Anil Kumar Singh and Sudheer Kolachina

Disambiguation of Preposition Sense Using Linguistically Motivated Features
Stephen Tratz and Dirk Hovy

Tuesday, June 2, 2009

Plenary Session

9:00–9:10 Paper Awards

9:10–9:40 *Unsupervised Morphological Segmentation with Log-Linear Models*
Hoifung Poon, Colin Cherry and Kristina Toutanova

9:40–10:10 *11,001 New Features for Statistical Machine Translation*
David Chiang, Kevin Knight and Wei Wang

10:10-10:40 **Break**

Session 4A: Machine Translation

10:10–10:35 *Efficient Parsing for Transducer Grammars*
John DeNero, Mohit Bansal, Adam Pauls and Dan Klein

10:35–10:50 *Preference Grammars: Softening Syntactic Constraints to Improve Statistical Machine Translation*
Ashish Venugopal, Andreas Zollmann, Noah A. Smith and Stephan Vogel

10:50–11:15 *Using a Dependency Parser to Improve SMT for Subject-Object-Verb Languages*
Peng Xu, Jaeho Kang, Michael Ringgaard and Franz Och

11:15–11:40 *Learning Bilingual Linguistic Reordering Model for Statistical Machine Translation*
Han-Bin Chen, Jian-Cheng Wu and Jason S. Chang

Session 4B: Sentiment Analysis / Information Extraction

10:10–10:35 *May All Your Wishes Come True: A Study of Wishes and How to Recognize Them*
Andrew B. Goldberg, Nathanael Fillmore, David Andrzejewski, Zhiting Xu, Bryan Gibson and Xiaojin Zhu

10:35–10:50 *Predicting Risk from Financial Reports with Regression*
Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi and Noah A. Smith

10:50–11:15 *Domain Adaptation with Latent Semantic Association for Named Entity Recognition*
Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, Xian Wu and Zhong Su

11:15–11:40 *Semi-Automatic Entity Set Refinement*
Vishnu Vyas and Patrick Pantel

Tuesday, June 2, 2009 (continued)

Session 4C: Machine Learning / Morphology and Phonology

10:10–10:35 *Unsupervised Constraint Driven Learning For Transliteration Discovery*
Ming-Wei Chang, Dan Goldwasser, Dan Roth and Yuancheng Tu

10:35–10:50 *On the Syllabification of Phonemes*
Susan Bartlett, Grzegorz Kondrak and Colin Cherry

10:50–11:15 *Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars*
Mark Johnson and Sharon Goldwater

11:15–11:40 No Presentation

12:20–2:00 **Lunch Break**

Session 5A: Short Paper Presentations: Machine Translation / Generation / Semantics

Note: all short papers are located in the Companion volume of the proceedings

2:00–2:15 *Statistical Post-Editing of a Rule-Based Machine Translation System*
Antonio-L. Lagarda, Vicent Alabau, Francisco Casacuberta, Roberto Silva and Enrique Daz-de-Liao

2:15–2:30 *On the Importance of Pivot Language Selection for Statistical Machine Translation*
Michael Paul, Hirofumi Yamamoto, Eiichiro Sumita and Satoshi Nakamura

2:30–2:45 *Tree Linearization in English: Improving Language Model Based Approaches*
Katja Filippova and Michael Strube

2:45–3:00 *Determining the position of adverbial phrases in English*
Huayan Zhong and Amanda Stent

3:00–3:15 *Estimating and Exploiting the Entropy of Sense Distributions*
Peng Jin, Diana McCarthy, Rob Koeling and John Carroll

3:15–3:30 *Semantic classification with WordNet Kernels*
Diarmuid Saghdha

Tuesday, June 2, 2009 (continued)

Session 5B: Short Paper Presentations: Machine Learning / Syntax

Note: all short papers are located in the Companion volume of the proceedings

2:00–2:15 *Sentence Boundary Detection and the Problem with the U.S.*
Dan Gillick

2:15–2:30 *Quadratic Features and Deep Architectures for Chunking*
Joseph Turian, James Bergstra and Yoshua Bengio

2:30–2:45 *Active Zipfian Sampling for Statistical Parser Training*
Onur Cobanoglu

2:45–3:00 *Combining Constituent Parsers*
Victoria Fossum and Kevin Knight

3:00–3:15 *Recognising the Predicate-argument Structure of Tagalog*
Meladel Mistica and Timothy Baldwin

3:15–3:30 *Reverse Revision and Linear Tree Combination for Dependency Parsing*
Giuseppe Attardi and Felice Dell'Orletta

Session 5C: Short Paper Presentations: SPECIAL SESSION – Speech Indexing and Retrieval

Note: all short papers are located in the Companion volume of the proceedings

2:00–2:15 *Introduction to the Special Session on Speech Indexing and Retrieval*

2:15–2:30 *Anchored Speech Recognition for Question Answering*
Sibel Yaman, Gokan Tur, Dimitra Vergyri, Dilek Hakkani-Tur, Mary Harper and Wen Wang

2:30–2:45 *Score Distribution Based Term Specific Thresholding for Spoken Term Detection*
Dogan Can and Murat Saraclar

2:45–3:00 *Automatic Chinese Abbreviation Generation Using Conditional Random Field*
Dong Yang, Yi-Cheng Pan and Sadaoki Furui

Tuesday, June 2, 2009 (continued)

3:00–3:15 *Fast decoding for open vocabulary spoken term detection*
Bhuvana Ramabhadran, Abhinav Sethy, Jonathan Mamou, Brian Kingsbury and Upendra Chaudhari

3:15–3:30 *Tightly coupling Speech Recognition and Search*
Taniya Mishra and Srinivas Bangalore

3:30–4:00 **Break**

Session 6A: Syntax and Parsing

4:00–4:25 *Joint Parsing and Named Entity Recognition*
Jenny Rose Finkel and Christopher D. Manning

4:25–4:50 *Minimal-length linearizations for mildly context-sensitive dependency trees*
Y. Albert Park and Roger Levy

4:50–5:15 *Positive Results for Parsing with a Bounded Stack using a Model-Based Right-Corner Transform*
William Schuler

Session 6B: Discourse and Summarization

4:00–4:25 *Hierarchical Text Segmentation from Multi-Scale Lexical Cohesion*
Jacob Eisenstein

4:25–4:50 *Exploring Content Models for Multi-Document Summarization*
Aria Haghighi and Lucy Vanderwende

4:50–5:15 *Global Models of Document Structure using Latent Permutations*
Harr Chen, S.R.K. Branavan, Regina Barzilay and David R. Karger

Tuesday, June 2, 2009 (continued)

Session 6C: Spoken Language Systems

4:00–4:25 *Assessing and Improving the Performance of Speech Recognition for Incremental Systems*
Timo Baumann, Michaela Atterer and David Schlangen

4:25–4:50 *Geo-Centric Language Models for Local Business Voice Search*
Amanda Stent, Ilija Zeljkovic, Diamantino Caseiro and Jay Wilpon

4:50–5:15 *Improving the Arabic Pronunciation Dictionary for Phone and Word Recognition with Linguistically-Based Pronunciation Rules*
Fadi Biadsy, Nizar Habash and Julia Hirschberg

Wednesday, June 3, 2009

Plenary Session

9:00–10:10 Invited Talk: *Ketchup, Espresso, and Chocolate Chip Cookies: Travels in the Language of Food*
Dan Jurafsky

10:10–10:40 **Break**

Session 7A: Machine Translation

10:40–11:05 *Using a maximum entropy model to build segmentation lattices for MT*
Chris Dyer

11:05–11:30 *Active Learning for Statistical Phrase-based Machine Translation*
Gholamreza Haffari, Maxim Roy and Anoop Sarkar

11:30–11:55 *Semi-Supervised Lexicon Mining from Parenthetical Expressions in Monolingual Web Pages*
Xianchao Wu, Naoaki Okazaki and Jun'ichi Tsujii

11:55–12:20 *Hierarchical Phrase-Based Translation with Weighted Finite State Transducers*
Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga and William Byrne

Wednesday, June 3, 2009 (continued)

Session 7B: Speech Recognition and Language Modeling

10:40–11:05 *Improved pronunciation features for construct-driven assessment of non-native spontaneous speech*
Lei Chen, Klaus Zechner and Xiaoming Xi

11:05–11:30 *Performance Prediction for Exponential Language Models*
Stanley Chen

11:30–11:55 *Tied-Mixture Language Modeling in Continuous Space*
Ruhi Sarikaya, Mohamed Afify and Brian Kingsbury

11:55–12:20 *Shrinking Exponential Language Models*
Stanley Chen

Session 7C: Sentiment Analysis

10:40–11:05 *Predicting Response to Political Blog Posts with Topic Models*
Tae Yano, William W. Cohen and Noah A. Smith

11:05–11:30 *An Iterative Reinforcement Approach for Fine-Grained Opinion Mining*
Weifu Du and Songbo Tan

11:30–11:55 *For a few dollars less: Identifying review pages sans human labels*
Luciano Barbosa, Ravi Kumar, Bo Pang and Andrew Tomkins

11:55–12:20 *More than Words: Syntactic Packaging and Implicit Sentiment*
Stephan Greene and Philip Resnik

12:20–1:40 **Lunch Break**

12:40-1:40 Panel Discussion: *Emerging Application Areas in Computational Linguistics*
Chaired by Bill Dolan, Microsoft
Panelists: Jill Burstein, Educational Testing Service; Joel Tetreault, Educational Testing Service; Patrick Pantel, Yahoo; Andy Hickl, Language Computer Corporation + Swingly

1:40–2:30 NAACL Business Meeting

Wednesday, June 3, 2009 (continued)

Session 8A: Large-scale NLP

2:30–2:55 *Streaming for large scale NLP: Language Modeling*
Amit Goyal, Hal Daume III and Suresh Venkatasubramanian

2:55–3:20 *The Effect of Corpus Size on Case Frame Acquisition for Discourse Analysis*
Ryohei Sasano, Daisuke Kawahara and Sadao Kurohashi

3:20–3:45 *Semantic-based Estimation of Term Informativeness*
Kirill Kireyev

Session 8B: Syntax and Parsing

2:30–2:55 *Optimal Reduction of Rule Length in Linear Context-Free Rewriting Systems*
Carlos Gómez-Rodríguez, Marco Kuhlmann, Giorgio Satta and David Weir

2:55–3:20 *Inducing Compact but Accurate Tree-Substitution Grammars*
Trevor Cohn, Sharon Goldwater and Phil Blunsom

3:20–3:45 *Hierarchical Search for Parsing*
Adam Pauls and Dan Klein

Session 8C: Discourse and Summarization

2:30–2:55 *An effective Discourse Parser that uses Rich Linguistic Information*
Rajen Subba and Barbara Di Eugenio

2:55–3:20 *Graph-Cut-Based Anaphoricity Determination for Coreference Resolution*
Vincent Ng

3:20–3:45 *Using Citations to Generate surveys of Scientific Paradigms*
Saif Mohammad, Bonnie Dorr, Melissa Egan, Ahmed Hassan, Pradeep Muthukrishan, Vahed Qazvinian, Dragomir Radev and David Zajic

3:45–4:15 **Break**

Wednesday, June 3, 2009 (continued)

Session 9A: Machine Learning

4:15–4:40 *Non-Parametric Bayesian Areal Linguistics*
Hal Daume III

4:40–5:05 *Hierarchical Bayesian Domain Adaptation*
Jenny Rose Finkel and Christopher D. Manning

5:05–5:30 *Online EM for Unsupervised Models*
Percy Liang and Dan Klein

Session 9B: Dialog Systems

4:15–4:40 *Unsupervised Approaches for Automatic Keyword Extraction Using Meeting Transcripts*
Feifan Liu, Deana Pennell, Fei Liu and Yang Liu

4:40–5:05 *A Finite-State Turn-Taking Model for Spoken Dialog Systems*
Antoine Raux and Maxine Eskenazi

5:05–5:30 *Extracting Social Meaning: Identifying Interactional Style in Spoken Conversation*
Dan Jurafsky, Rajesh Ranganath and Dan McFarland

Session 9C: Syntax and Parsing

4:15–4:40 *Linear Complexity Context-Free Parsing Pipelines via Chart Constraints*
Brian Roark and Kristy Hollingshead

4:40–5:05 *Improved Syntactic Models for Parsing Speech with Repairs*
Tim Miller

5:05–5:30 *A model of local coherence effects in human sentence processing as consequences of updates from bottom-up prior to posterior beliefs*
Klinton Bicknell and Roger Levy

Subjectivity Recognition on Word Senses via Semi-supervised Mincuts

Fangzhong Su
School of Computing
University of Leeds
fzsu@comp.leeds.ac.uk

Katja Markert
School of Computing
University of Leeds
markert@comp.leeds.ac.uk

Abstract

We supplement WordNet entries with information on the subjectivity of its word senses. Supervised classifiers that operate on word sense definitions in the same way that text classifiers operate on web or newspaper texts need large amounts of training data. The resulting data sparseness problem is aggravated by the fact that dictionary definitions are very short. We propose a semi-supervised minimum cut framework that makes use of both WordNet definitions and its relation structure. The experimental results show that it outperforms supervised minimum cut as well as standard supervised, non-graph classification, reducing the error rate by 40%. In addition, the semi-supervised approach achieves the same results as the supervised framework with less than 20% of the training data.

1 Introduction

There is considerable academic and commercial interest in processing subjective content in text, where subjective content refers to any expression of a private state such as an opinion or belief (Wiebe et al., 2005). Important strands of work include the identification of subjective content and the determination of its *polarity*, i.e. whether a favourable or unfavourable opinion is expressed.

Automatic identification of subjective content often relies on word indicators, such as unigrams (Pang et al., 2002) or predetermined sentiment lexica (Wilson et al., 2005). Thus, the word *positive* in the sentence *"This deal is a positive development for our company."* gives a strong indication that the sentence contains a favourable opinion. However, such word-based indicators can be misleading for two reasons. First, contextual indicators such as irony and negation can reverse subjectivity or polarity indications (Polanyi and Zaenen, 2004). Second, different word senses of a single word can actually be of different subjectivity or polarity. A typical *subjectivity-ambiguous* word, i.e. a word that has at least one subjective and at least one objective sense, is *positive*, as shown by the two example senses given below.[1]

(1) positive, electropositive—having a positive electric charge;"protons are positive" (*objective*)

(2) plus, positive—involving advantage or good; "a plus (or positive) factor" (*subjective*)

We concentrate on this latter problem by automatically creating lists of subjective senses, instead of subjective words, via adding subjectivity labels for senses to electronic lexica, using the example of WordNet. This is important as the problem of subjectivity-ambiguity is frequent: We (Su and Markert, 2008) find that over 30% of words in our dataset are subjectivity-ambiguous. Information on subjectivity of senses can also improve other tasks such as word sense disambiguation (Wiebe and Mihalcea, 2006). Moreover, Andreevskaia and Bergler (2006) show that the performance of automatic annotation of subjectivity at the *word* level can be hurt by the presence of subjectivity-ambiguous words in the training sets they use.

[1] All examples in this paper are from WordNet 2.0.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 1–9,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

We propose a semi-supervised approach based on minimum cut in a lexical relation graph to assign subjectivity (subjective/objective) labels to word senses.[2] Our algorithm outperforms supervised minimum cuts and standard supervised, non-graph classification algorithms (like SVM), reducing the error rate by up to 40%. In addition, the semi-supervised approach achieves the same results as the supervised framework with less than 20% of the training data. Our approach also outperforms prior approaches to the subjectivity recognition of word senses and performs well across two different data sets.

The remainder of this paper is organized as follows. Section 2 discusses previous work. Section 3 describes our proposed semi-supervised minimum cut framework in detail. Section 4 presents the experimental results and evaluation, followed by conclusions and future work in Section 5.

2 Related Work

There has been a large and diverse body of research in opinion mining, with most research at the text (Pang et al., 2002; Pang and Lee, 2004; Popescu and Etzioni, 2005; Ounis et al., 2006), sentence (Kim and Hovy, 2005; Kudo and Matsumoto, 2004; Riloff et al., 2003; Yu and Hatzivassiloglou, 2003) or word (Hatzivassiloglou and McKeown, 1997; Turney and Littman, 2003; Kim and Hovy, 2004; Takamura et al., 2005; Andreevskaia and Bergler, 2006; Kaji and Kitsuregawa, 2007) level. An up-to-date overview is given in Pang and Lee (2008).

Graph-based algorithms for classification into subjective/objective or positive/negative language units have been mostly used at the sentence and document level (Pang and Lee, 2004; Agarwal and Bhattacharyya, 2005; Thomas et al., 2006), instead of aiming at dictionary annotation as we do. We also cannot use prior graph construction methods for the document level (such as physical proximity of sentences, used in Pang and Lee (2004)) at the word sense level. At the word level Takamura et al. (2005) use a semi-supervised spin model for word polarity determination, where the graph is constructed using a variety of information such as gloss co-occurrences and WordNet links. Apart from using a different graph-based model from ours, they assume that subjectivity recognition has already been achieved prior to polarity recognition and test against word lists containing subjective words only. However, Kim and Hovy (2004) and Andreevskaia and Bergler (2006) show that subjectivity recognition might be the harder problem with lower human agreement and automatic performance. In addition, we deal with classification at the *word sense* level, treating also subjectivity-ambiguous words, which goes beyond the work in Takamura et al. (2005).

Word Sense Level: There are three prior approaches addressing *word sense* subjectivity or polarity classification. Esuli and Sebastiani (2006) determine the polarity (positive/negative/objective) of word senses in WordNet. However, there is no evaluation as to the accuracy of their approach. They then extend their work (Esuli and Sebastiani, 2007) by applying the Page Rank algorithm to rank the WordNet senses in terms of how strongly a sense possesses a given semantic property (e.g., positive or negative). Apart from us tackling subjectivity instead of polarity, their Page Rank graph is also constructed focusing on WordNet glosses (linking glosses containing the same words), whereas we concentrate on the use of WordNet relations.

Both Wiebe and Mihalcea (2006) and our prior work (Su and Markert, 2008) present an annotation scheme for word sense subjectivity and algorithms for automatic classification. Wiebe and Mihalcea (2006) use an algorithm relying on distributional similarity and an independent, large manually annotated opinion corpus (MPQA) (Wiebe et al., 2005). One of the disadvantages of their algorithm is that it is restricted to senses that have distributionally similar words in the MPQA corpus, excluding 23% of their test data from automatic classification. Su and Markert (2008) present supervised classifiers, which rely mostly on WordNet glosses and do not effectively exploit WordNet's relation structure.

3 Semi-Supervised Mincuts

3.1 Minimum Cuts: The Main Idea

Binary classification with minimum cuts (Mincuts) in graphs is based on the idea that similar items

[2] It can be argued that subjectivity labels are maybe rather more graded than the clear-cut binary distinction we assign. However, in Su and Markert (2008a) as well as Wiebe and Mihalcea (2006) we find that human can assign the binary distinction to word senses with a high level of reliability.

should be grouped in the same cut. All items in the training/test data are seen as vertices in a graph with undirected weighted edges between them specifying how strong the similarity/association between two vertices is. We use minimum s-t cuts: the graph contains two particular vertices s (source, corresponds to subjective) and t (sink, corresponds to objective) and each vertex u is connected to s and t via a weighted edge that can express how likely u is to be classified as s or t in isolation.

Binary classification of the vertices is equivalent to splitting the graph into two disconnected subsets of all vertices, S and T with $s \in S$ and $t \in T$. This corresponds to removing a set of edges from the graph. As similar items should be in the same part of the split, the best split is one which removes edges with low weights. In other words, a minimum cut problem is to find a partition of the graph which minimizes the following formula, where $w(u, v)$ expresses the weight of an edge between two vertices.

$$W(S,T) = \sum_{u \in S, v \in T} w(u,v)$$

Globally optimal minimum cuts can be found in polynomial time and near-linear running time in practice, using the maximum flow algorithm (Pang and Lee, 2004; Cormen et al., 2002).

3.2 Why might Semi-supervised Minimum Cuts Work?

We propose semi-supervised mincuts for subjectivity recognition on senses for several reasons.

First, our problem satisfies two major conditions necessary for using minimum cuts. It is a binary classification problem (subjective vs. objective senses) as is needed to divide the graph into two components. Our dataset also lends itself naturally to s-t Mincuts as we have two different views on the data. Thus, the edges of a vertex (=sense) to the source/sink can be seen as the probability of a sense being subjective or objective without taking similarity to other senses into account, for example via considering only the sense gloss. In contrast, the edges between two senses can incorporate the WordNet relation hierarchy, which is a good source of similarity for our problem as many WordNet relations are *subjectivity-preserving*, i.e. if two senses are connected via such a relation they are likely to be both subjective or both objective.[3] An example here is the antonym relation, where two antonyms such as *good—morally admirable* and *evil, wicked—morally bad or wrong* are both subjective.

Second, Mincuts can be easily expanded into a semi-supervised framework (Blum and Chawla, 2001). This is essential as the existing labeled datasets for our problem are small. In addition, glosses are short, leading to sparse high dimensional vectors in standard feature representations. Also, WordNet connections between different parts of the WordNet hierarchy can also be sparse, leading to relatively isolated senses in a graph in a supervised framework. Semi-supervised Mincuts allow us to import unlabeled data that can serve as bridges to isolated components. More importantly, as the unlabeled data can be chosen to be related to the labeled and test data, they might help pull test data to the right cuts (categories).

3.3 Formulation of Semi-supervised Mincuts

The formulation of our semi-supervised Mincut for sense subjectivity classification involves the following steps, which we later describe in more detail.

1. We define two vertices s (source) and t (sink), which correspond to the "subjective" and "objective" category, respectively. Following the definition in Blum and Chawla (2001), we call the vertices s and t *classification vertices*, and all other vertices (labeled, test, and unlabeled data) *example vertices*. Each example vertex corresponds to one WordNet sense and is connected to both s and t via a weighted edge. The latter guarantees that the graph is connected.
2. For the test and unlabeled examples, we see the edges to the classification vertices as the probability of them being subjective/objective disregarding other example vertices. We use a supervised classifier to set these edge weights. For the labeled training examples, they are connected by edges with a high constant weight to the classification vertices that they belong to.
3. WordNet relations are used to construct the edges *between two example vertices*. Such

[3] See Kamps et al. (2004) for an early indication of such properties for some WordNet relations.

edges can exist between any pair of example vertices, for example between two unlabeled examples.

4. After graph construction we then employ a maximum-flow algorithm to find the minimum s-t cuts of the graph. The cut in which the source vertex s lies is classified as "subjective", and the cut in which the sink vertex t lies is "objective".

We now describe the above steps in more detail.

Selection of unlabeled data: Random selection of unlabeled data might hurt the performance of Mincuts, as they might not be related to any sense in our training/test data (denoted by A). Thus a basic principle is that the selected unlabeled senses should be related to the training/test data by WordNet relations. We therefore simply scan each sense in A, and collect all senses related to it via one of the WordNet relations in Table 1. All such senses that are not in A are collected in the unlabeled data set.

Weighting of edges to the classification vertices: The edge weight to s and t represents how likely it is that an example vertex is initially put in the cut in which s (subjective) or t (objective) lies. For unlabeled and test vertices, we use a supervised classifier (SVM[4]) with the labeled data as training data to assign the edge weights. The SVM is also used as a baseline and its features are described in Section 4.3. As we do not wish the Mincut to reverse labels of the labeled training data, we assign a high constant weight of 5 to the edge between a labeled vertex and its corresponding classification vertex, and a low weight of 0.01 to the edge to the other classification vertex.

Assigning weights to WordNet relations: We connect two vertices that are linked by one of the ten WordNet relations in Table 1 via an edge. Not all WordNet relations we use are subjectivity-preserving to the same degree: for example, hyponyms (such as *simpleton*) of objective senses (such as *person*) do not have to be objective. However, we aim for high graph connectivity and we can assign different weights to different relations to reflect the degree to which they are subjectivity-preserving. Therefore, we experiment with two methods of weight assignment. Method 1 (NoSL) assigns the same constant weight of 1.0 to all WordNet relations.

Method 2 (SL) reflects different degrees of preserving subjectivity. To do this, we adapt an unsupervised method of generating a large noisy set of subjective and objective senses from our previous work (Su and Markert, 2008). This method uses a list of subjective words (SL)[5] to classify each WordNet sense with at least two subjective words in its gloss as subjective and all other senses as objective. We then count how often two senses related via a given relation have the same or a different subjectivity label. The weight is computed by #same/(#same+#different). Results are listed in Table 1.

Table 1: Relation weights (Method 2)

Method	#Same	#Different	Weight
Antonym	2,808	309	0.90
Similar-to	6,887	1,614	0.81
Derived-from	4,630	947	0.83
Direct-Hypernym	71,915	8,600	0.89
Direct-Hyponym	71,915	8,600	0.89
Attribute	350	109	0.76
Also-see	1,037	337	0.75
Extended-Antonym	6,917	1,651	0.81
Domain	4,387	892	0.83
Domain-member	4,387	892	0.83

Example graph: An example graph is shown in Figure 1. The three example vertices correspond to the senses **religious**—*extremely scrupulous and conscientious,* **scrupulous**—*having scruples; arising from a sense of right and wrong; principled*; and **flicker**, *spark, glint—a momentary flash of light* respectively. The vertex "scrupulous" is unlabeled data derived from the vertex "religious"(a test item) by the relation "similar-to".

4 Experiments and Evaluation

4.1 Datasets

We conduct the experiments on two different gold standard datasets. One is the Micro-WNOp corpus,

[4]We employ LIBSVM, available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`. Linear kernel and probability estimates are used in this work.

[5]Available at `http://www.cs.pitt.edu/mpqa`

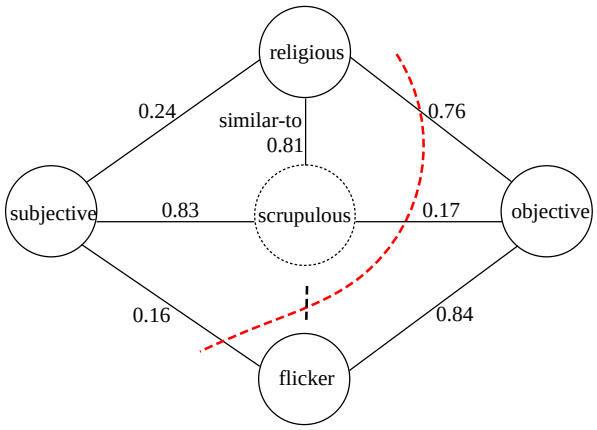


Figure 1: Graph of Word Senses

which is representative of the part-of-speech distribution in WordNet [6]. It includes 298 words with 703 objective and 358 subjective WordNet senses. The second one is the dataset created by Wiebe and Mihalcea (2006).[7] It only contains noun and verb senses, and includes 60 words with 236 objective and 92 subjective WordNet senses. As the Micro-WNOp set is larger and also contains adjective and adverb senses, we describe our results in more detail on that corpus in the Section 4.3 and 4.4. In Section 4.5, we shortly discuss results on Wiebe&Mihalcea's dataset.

4.2 Baseline and Evaluation

We compare to a baseline that assigns the most frequent category *objective* to all senses, which achieves an accuracy of 66.3% and 72.0% on Micro-WNOp and Wiebe&Mihalcea's dataset respectively. We use the McNemar test at the significance level of 5% for significance statements. All evaluations are carried out by 10-fold cross-validation.

4.3 Standard Supervised Learning

We use an SVM classifier to compare our proposed semi-supervised Mincut approach to a reasonable baseline.[8] Three different feature types are used.

Lexical Features (L): a bag-of-words representation of the sense glosses with stop word filtering.

Relation Features (R): First, we use two features for each of the ten WordNet relations in Table 1, describing how many relations of that type the sense has to senses in the subjective or objective part of the training set, respectively. This provides a non-graph summary of subjectivity-preserving links. Second, we manually collected a small set (denoted by $SubjSet$) of seven subjective verb and noun senses which are close to the root in WordNet's hypernym tree. A typical example element of $SubjSet$ is *psychological feature —a feature of the mental life of a living organism*, which indicates subjectivity for its hyponyms such as *hope — the general feeling that some desire will be fulfilled.* A binary feature describes whether a noun/verb sense is a hyponym of an element of $SubjSet$.

Monosemous Feature (M): for each sense, we scan if a monosemous word is part of its synset. If so, we further check if the monosemous word is collected in the subjective word list (SL). The intuition is that if a monosemous word is subjective, obviously its (single) sense is subjective. For example, the sense *uncompromising, inflexible—not making concessions* is subjective, as "uncompromising" is a monosemous word and also in SL.

We experiment with different combinations of features and the results are listed in Table 2, prefixed by "SVM". All combinations perform significantly better than the more frequent category baseline and similarly to the supervised Naive Bayes classifier (see S&M in Table 2) we used in Su and Markert (2008). However, improvements by adding more features remain small.

In addition, we compare to a supervised classifier (see Lesk in Table 2) that just assigns each sense the subjectivity label of its most similar sense in the training data, using Lesk's similarity measure from Pedersen's WordNet similarity package[9]. We use Lesk as it is one of the few measures applicable across all parts-of-speech.

[6] Available at `http://www.comp.leeds.ac.uk/markert/data`. This dataset was first used with a different annotation scheme in Esuli and Sebastiani (2007) and we also used it in Su and Markert (2008).

[7] Available at `http://www.cs.pitt.edu/~wiebe/pubs/papers/goldstandard.total.acl06`.

[8] This SVM is also used to provide the edge weights to the *classification vertices* in the Mincut approach.

[9] Available at `http://www.d.umn.edu/~tpederse/similarity.html`.

Table 2: Results of SVM and Mincuts with different settings of feature

Method	Subjective			Objective			Accuracy
	Precision	Recall	F-score	Precision	Recall	F-score	
Baseline	N/A	0	N/A	66.3%	**100%**	79.7%	66.3%
S&M	66.2%	64.5%	65.3%	82.2%	83.2%	82.7%	76.9%
Lesk	65.6%	50.3%	56.9%	77.5%	86.6%	81.8%	74.4%
SVM-L	69.6%	37.7%	48.9%	74.3%	91.6%	82.0%	73.4%
L-SL	82.0%	43.3%	56.7%	76.7%	95.2%	85.0%	77.7%
L-NoSL	80.8%	43.6%	56.6%	76.7%	94.7%	84.8%	77.5%
SVM-LM	68.9%	42.2%	52.3%	75.4%	90.3%	82.2%	74.1%
LM-SL	83.2%	44.4%	57.9%	77.1%	95.4%	85.3%	78.2%
LM-NoSL	83.6%	44.1%	57.8%	77.1%	95.6%	85.3%	78.2%
SVM-LR	68.4%	45.3%	54.5%	76.2%	89.3%	82.3%	74.5%
LR-SL	82.7%	65.4%	73.0%	84.1%	93.0%	88.3%	83.7%
LR-NoSL	82.4%	65.4%	72.9%	84.0%	92.9%	88.2%	83.6%
SVM-LRM	69.8%	47.2%	56.3%	76.9%	89.6%	82.8%	75.3%
LRM-SL	**85.5%**	65.6%	**74.2%**	**84.4%**	94.3%	**89.1%**	**84.6%**
LRM-NoSL	84.6%	**65.9%**	74.1%	**84.4%**	93.9%	88.9%	84.4%

[1] L, R and M correspond to the lexical, relation and monosemous features respectively.

[2] SVM-L corresponds to using lexical features only for the SVM classifier. Likewise, SVM-LRM corresponds to using a combination for lexical, relation, and monosemous features for the SVM classifier.

[3] L-SL corresponds to the Mincut that uses only lexical features for the SVM classifier, and subjective list (SL) to infer the weight of WordNet relations. Likewise, LM-NoSL corresponds to the Mincut algorithm that uses lexical and monosemous features for the SVM, and predefined constants for WordNet relations (without subjective list).

4.4 Semi-supervised Graph Mincuts

Using our formulation in Section 3.3, we import 3,220 senses linked by the ten WordNet relations to any senses in Micro-WNOp as unlabeled data. We construct edge weights to classification vertices using the SVM discussed above and use WordNet relations for links between example vertices, weighted by either constants (NoSL) or via the method illustrated in Table 1 (SL). The results are also summarized in Table 2. Semi-supervised Mincuts always significantly outperform the corresponding SVM classifiers, regardless of whether the subjectivity list is used for setting edge weights. We can also see that we achieve good results without using any other knowledge sources (setting LR-NoSL).

The example in Figure 1 explains why semi-supervised Mincuts outperforms the supervised approach. The vertex "religious" is initially assigned the subjective/objective probabilities 0.24/0.76 by the SVM classifier, leading to a wrong classification. However, in our graph-based Mincut framework, the vertex "religious" might link to other vertices (for example, it links to the vertex "scrupulous" in the unlabeled data by the relation "similar-to"). The mincut algorithm will put vertices "religious" and "scrupulous" in the same cut (subjective category) as this results in the least cost 0.93 (ignoring the cost of assigning the unrelated sense of "flicker"). In other words, the edges between the vertices are likely to correct some initially *wrong* classification and pull the vertices into the *right* cuts.

In the following we will analyze the best minimum cut algorithm LRM-SL in more detail. We measure its accuracy for each part-of-speech in the Micro-WNOp dataset. The number of noun, adjective, adverb and verb senses in Micro-WNOp is 484, 265, 31 and 281, respectively. The result is listed in Table 3. The significantly better performance of semi-supervised mincuts holds across all parts-of-speech but the small set of adverbs, where there is no significant difference between the baseline, SVM and the Mincut algorithm.

Table 3: Accuracy for Different Part-Of-Speech

Method	Noun	Adjective	Adverb	Verb
Baseline	76.9%	61.1%	77.4%	72.6%
SVM	81.4%	63.4%	**83.9%**	75.1%
Mincut	**88.6%**	**78.9%**	77.4%	**84.0%**

We will now investigate how LRM-SL performs with different sizes of labeled and unlabeled data. All learning curves are generated via averaging 10 learning curves from 10-fold cross-validation.

Performance with different sizes of labeled data: we randomly generate subsets of labeled data A_1, A_2... A_n, and guarantee that $A_1 \subset A_2 ... \subset A_n$. Results for the best SVM (LRM) and the best minimum cut (LRM-SL) are listed in Table 4, and the corresponding learning curve is shown in Figure 2. As can be seen, the semi-supervised Mincuts is consistently better than SVM. Moreover, the semi-supervised Mincut with only 200 labeled data items performs even better than SVM with 954 training items (78.9% *vs* 75.3%), showing that our semi-supervised framework allows for a training data reduction of more than 80%.

Table 4: Accuracy with different sizes of labeled data

# labeled data	SVM	Mincuts
100	69.1%	72.2%
200	72.6%	78.9%
400	74.4%	82.7%
600	75.5%	83.7%
800	76.0%	84.1%
900	75.6%	84.8%
954 (all)	75.3%	84.6%

Performance with different sizes of unlabeled data: We propose two different settings.

Option1: Use a subset of the ten relations to generate the unlabeled data (and edges between example vertices). For example, we first use {antonym, similar-to} only to obtain a unlabeled dataset U_1, then use a larger subset of the relations like {antonym, similar-to, direct-hyponym, direct-hypernym} to generate another unlabeled dataset U_2, and so forth. Obviously, U_i is a subset of U_{i+1}.

Option2: Use all the ten relations to generate the unlabeled data U. We then randomly select subsets of U, such as subset U_1, U_2 and U_3, and guarantee that $U_1 \subset U_2 \subset U_3 \subset \ldots U$.

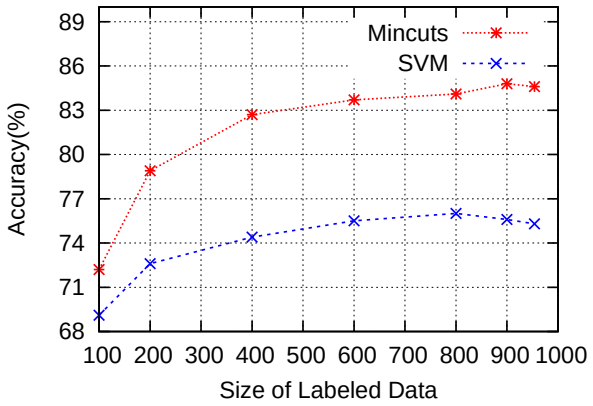


Figure 2: Learning curve with different sizes of labeled data

The results are listed in Table 5 and Table 6 respectively. The corresponding learning curves are shown in Figure 3. We see that performance improves with the increase of unlabeled data. In addition, the curves seem to converge when the size of unlabeled data is larger than 3,000. From the results in Tabel 5 one can also see that hyponymy is the relation accounting for the largest increase.

Table 6: Accuracy with different sizes of unlabeled data (random selection)

# unlabeled data	Accuracy
0	75.9%
200	76.5%
500	78.6%
1000	80.2%
2000	82.8%
3000	84.0%
3220	84.6%

Furthermore, these results also show that a supervised mincut without unlabeled data performs only on a par with other supervised classifiers (75.9%). The reason is that if we exclude the unlabeled data, there are only 67 WordNet relations/edges between senses in the small Micro-WNOp dataset. In contrast, the use of unlabeled data adds more edges (4,586) to the graph, which strongly affects the graph cut partition (see also Figure 1).

4.5 Comparison to Prior Approaches

In our previous work (Su and Markert, 2008), we report 76.9% as the best accuracy on the same Micro-

Table 5: Accuracy with different sizes of unlabeled data from WordNet relation

Relation	# unlabeled data	Accuracy
{∅}	0	75.3%
{similar-to}	418	79.1%
{similar-to, antonym}	514	79.5%
{similar-to, antonym, direct-hypernym, direct-hyponym}	2,721	84.4%
{similar-to, antonym, direct-hypernym, direct-hyponym, also-see, extended-antonym}	3,004	84.4%
{similar-to, antonym, direct-hypernym, direct-hyponym, also-see, extended-antonym, derived-from, attribute, domain, domain-member}	3,220	84.6%

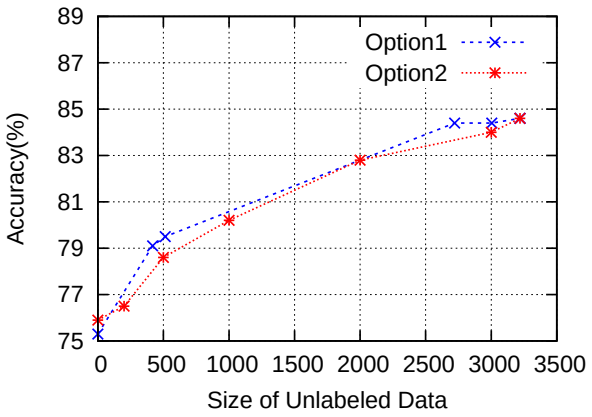


Figure 3: Learning curve with different sizes of unlabeled data

WNOp dataset used in the previous sections, using a supervised Naive Bayes (S&M in Tabel 2). Our best result from Mincuts is significantly better at 84.6% (see LRM-SL in Table 2).

For comparison to Wiebe and Mihalcea (2006), we use their dataset for testing, henceforth called `Wiebe` (see Section 4.1 for a description). Wiebe and Mihalcea (2006) report their results in precision and recall curves for subjective senses, such as a precision of about 55% at a recall of 50% for subjective senses. Their F-score for subjective senses seems to remain relatively static at 0.52 throughout their precision/recall curve.

We run our best Mincut LRM-SL algorithm with two different settings on `Wiebe`. Using Micro-WNOp as training set and `Wiebe` as test set, we achieve an accuracy of 83.2%, which is similar to the results on the Micro-WNOp dataset. At the recall of 50% we achieve a precision of 83.6% (in comparison to their precision of 55% at the same recall). Our F-score is 0.63 (vs. 0.52).

To check whether the high performance is just due to our larger training set, we also conduct 10-fold cross-validation on `Wiebe`. The accuracy achieved is 81.1% and the F-score 0.56 (vs. 0.52), suggesting that our algorithm performs better. Our algorithm can be used on all WordNet senses whereas theirs is restricted to senses that have distributionally similar words in the MPQA corpus (see Section 2). However, they use an unsupervised algorithm i.e. they do not need labeled word senses, although they do need a large, manually annotated opinion corpus.

5 Conclusion and Future Work

We propose a semi-supervised minimum cut algorithm for subjectivity recognition on word senses. The experimental results show that our proposed approach is significantly better than a standard supervised classification framework as well as a supervised Mincut. Overall, we achieve a 40% reduction in error rates (from an error rate of about 25% to an error rate of 15%). To achieve the results of standard supervised approaches with our model, we need less than 20% of their training data. In addition, we compare our algorithm to previous state-of-the-art approaches, showing that our model performs better on the same datasets.

Future work will explore other graph construction methods, such as the use of morphological relations as well as thesaurus and distributional similarity measures. We will also explore other semi-supervised algorithms.

References

Alekh Agarwal and Pushpak Bhattacharyya. 2005. Sentiment Analysis: A new Approach for Effective Use of Linguistic Knowledge and Exploiting Similarities in a Set of Documents to be Classified. *Proceedings of ICON'05*.

Alina Andreevskaia and Sabine Bergler. 2006. Mining WordNet for Fuzzy Sentiment: Sentiment Tag Extraction from WordNet Glosses. *Proceedings of EACL'06*.

Avrim Blum and Shuchi Chawla. 2001. Learning from Labeled and Unlabeled Data using Graph Mincuts. *Proceedings of ICML'01*.

Thomas Cormen, Charles Leiserson, Ronald Rivest and Clifford Stein. 2002. Introduction to Algorithms. *Second Edition, the MIT Press*.

Kushal Dave, Steve Lawrence, and David Pennock. 2003. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. *Proceedings of WWW'03*.

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. *Proceedings of LREC'06*.

Andrea Esuli and Fabrizio Sebastiani. 2007. PageRanking WordNet Synsets: An application to Opinion Mining. *Proceedings of ACL'07*.

Vasileios Hatzivassiloglou and Kathleen McKeown. 1997. Predicting the Semantic Orientation of Adjectives. *Proceedings of ACL'97*.

Nobuhiro Kaji and Masaru Kitsuregawa. 2007. Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents. *Proceedings of EMNLP'07*.

Japp Kamps, Maarten Marx, Robert Mokken, and Maarten de Rijke. 2004. Using WordNet to Measure Semantic Orientation of Adjectives. *Proceedings of LREC'04*.

Soo-Min Kim and Eduard Hovy. 2004. Determining the Sentiment of Opinions. *Proceedings of COLING'04*.

Soo-Min Kim and Eduard Hovy. 2005. Automatic Detection of Opinion Bearing Words and Sentences. *Proceedings of ICJNLP'05*.

Taku Kudo and Yuji Matsumoto. 2004. A Boosting Algorithm for Classification of Semi-structured Text. *Proceedings of EMNLP'04*.

Iadh Ounis, Maarten de Rijke, Craig Macdonald, Gilad Mishne and Ian Soboroff. 2006. Overview of the TREC-2006 Blog Track. *Proceedings of TREC'06*.

Bo Pang and Lillian Lee. 2004. A Sentiment Education: Sentiment Analysis Using Subjectivity summarization Based on Minimum Cuts. *Proceedings of ACL'04*.

Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval 2(1-2)*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. *Proceedings of EMNLP'02*.

Livia Polanyi and Annie Zaenen. 2004. Contextual Valence Shifters. *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting Product Features and Opinions from Reviews *Proceedings of EMNLP'05*.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning Subjective Nouns using Extraction Pattern Bootstrapping. *Proceedings of CoNLL'03*

Fangzhong Su and Katja Markert. 2008. From Words to Senses: A Case Study in Subjectivity Recognition. *Proceedings of COLING'08*.

Fangzhong Su and Katja Markert. 2008a. Eliciting Subjectivity and Polarity Judgements on Word Senses. *Proceedings of COLING'08 workshop of Human Judgements in Computational Linguistics*.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting Semantic Orientations of Words using Spin Model. *Proceedings of ACL'05*.

Matt Thomas, Bo Pang and Lilian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. *Proceedings of EMNLP'06*.

Peter Turney. 2002. Thumbs up or Thumbs down? Semantic orientation applied to unsupervised classification of reviews. *Proceedings of ACL'02*.

Peter Turney and Michael Littman. 2003. Measuring Praise and Criticism: Inference of Semantic Orientation from Association. *ACM Transaction on Information Systems*.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences. *Proceedings of EMNLP'03*.

Janyce Wiebe and Rada Micalcea. 2006. Word Sense and Subjectivity. *Proceedings of ACL'06*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating Expressions of Opinions and Emotions in Language. *Language Resources and Evaluation*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. *Proceedings of HLT/EMNLP'05*.

Integrating Knowledge for Subjectivity Sense Labeling

Yaw Gyamfi and **Janyce Wiebe**
University of Pittsburgh
{anti,wiebe}@cs.pitt.edu

Rada Mihalcea
University of North Texas
rada@cs.unt.edu

Cem Akkaya
University of Pittsburgh
cem@cs.pitt.edu

Abstract

This paper introduces an integrative approach to automatic word sense subjectivity annotation. We use features that exploit the hierarchical structure and domain information in lexical resources such as WordNet, as well as other types of features that measure the similarity of glosses and the overlap among sets of semantically related words. Integrated in a machine learning framework, the entire set of features is found to give better results than any individual type of feature.

1 Introduction

Automatic extraction of opinions, emotions, and sentiments in text (*subjectivity analysis*) to support applications such as product review mining, summarization, question answering, and information extraction is an active area of research in NLP.

Many approaches to opinion, sentiment, and subjectivity analysis rely on lexicons of words that may be used to express subjectivity. However, words may have both subjective and objective senses, which is a source of ambiguity in subjectivity and sentiment analysis. We show that even words judged in previous work to be reliable clues of subjectivity have significant degrees of subjectivity sense ambiguity.

To address this ambiguity, we present a method for automatically assigning subjectivity labels to word senses in a taxonomy, which uses new features and integrates more diverse types of knowledge than in previous work. We focus on nouns, which are challenging and have received less attention in automatic subjectivity and sentiment analysis.

A common approach to building lexicons for subjectivity analysis is to begin with a small set of seeds which are prototypically subjective (or positive/negative, in sentiment analysis), and then follow semantic links in WordNet-like resources. By far, the emphasis has been on horizontal relations, such as *synonymy* and *antonymy*. Exploiting vertical links opens the door to taking into account the information content of ancestor concepts of senses with known and unknown subjectivity. We develop novel features that measure the similarity of a target word sense with a seed set of senses known to be subjective, where the similarity between two concepts is determined by the extent to which they share information, measured by the information content associated with their least common subsumer (LCS). Further, particularizing the LCS features to domain greatly reduces calculation while still maintaining effective features.

We find that our new features do lead to significant improvements over methods proposed in previous work, and that the combination of all features gives significantly better performance than any single type of feature alone.

We also ask, given that there are many approaches to finding subjective words, if it would make sense for word- and sense-level approaches to work in tandem, or should we best view them as competing approaches? We give evidence suggesting that first identifying subjective words and then disambiguating their senses would be an effective approach to subjectivity sense labeling.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 10–18,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

There are several motivations for assigning subjectivity labels to senses. First, (Wiebe and Mihalcea, 2006) provide evidence that word sense labels, together with contextual subjectivity analysis, can be exploited to improve performance in word sense disambiguation. Similarly, given subjectivity sense labels, word-sense disambiguation may potentially help contextual subjectivity analysis. In addition, as lexical resources such as WordNet are developed further, subjectivity labels would provide principled criteria for refining word senses, as well as for clustering similar meanings to create more course-grained sense inventories.

For many opinion mining applications, polarity (positive, negative) is also important. The overall framework we envision is a layered approach: classifying instances as objective or subjective, and further classifying the subjective instances by polarity. Decomposing the problem into subproblems has been found to be effective for opinion mining. This paper addresses the first of these subproblems.

2 Background

We adopt the definitions of *subjective* and *objective* from Wiebe and Mihalcea (2006) (hereafter *WM*). Subjective expressions are words and phrases being used to express opinions, emotions, speculations, etc. WM give the following examples:

His **alarm** grew.
He **absorbed** the information quickly.
UCC/Disciples leaders **roundly condemned** the Iranian President's **verbal assault** on Israel.
What's the catch?

Polarity (also called *semantic orientation*) is also important to NLP applications in sentiment analysis and opinion extraction. In review mining, for example, we want to know whether an opinion about a product is positive or negative. Even so, we believe there are strong motivations for a separate subjective/objective (S/O) classification as well.

First, expressions may be subjective but not have any particular polarity. An example given by (Wilson et al., 2005) is *Jerome says the hospital* **feels** *no different than a hospital in the states.* An NLP application system may want to find a wide range of private states attributed to a person, such as their motivations, thoughts, and speculations, in addition to their positive and negative sentiments.

Second, distinguishing *S* and *O* instances has often proven more difficult than subsequent polarity classification. Researchers have found this at various levels of analysis, including the manual annotation of phrases (Takamura et al., 2006), sentiment classification of phrases (Wilson et al., 2005), sentiment tagging of words (Andreevskaia and Bergler, 2006b), and sentiment tagging of word senses (Esuli and Sebastiani, 2006a). Thus, effective methods for *S*/*O* classification promise to improve performance for sentiment classification. In fact, researchers in sentiment analysis have realized benefits by decomposing the problem into *S*/*O* and polarity classification (Yu and Hatzivassiloglou, 2003; Pang and Lee, 2004; Wilson et al., 2005; Kim and Hovy, 2006). One reason is that different features may be relevant for the two subproblems. For example, negation features are more important for polarity classification than for subjectivity classification.

Note that some of our features require vertical links that are present in WordNet for nouns and verbs but not for other parts of speech. Thus we address nouns (leaving verbs to future work). There are other motivations for focusing on nouns. Relatively little work in subjectivity and sentiment analysis has focused on subjective nouns. Also, a study (Bruce and Wiebe, 1999) showed that, of the major parts of speech, nouns are the most ambiguous with respect to the subjectivity of their instances.

Turning to word senses, we adopt the definitions from WM. First, subjective: "Classifying a sense as *S* means that, when the sense is used in a text or conversation, we expect it to express subjectivity; we also expect the phrase or sentence containing it to be subjective [WM, pp. 2-3]."

In WM, it is noted that sentences containing objective senses may not be objective, as in the sentence *Will someone shut that darn* **alarm** *off?* Thus, objective senses are defined as follows: "Classifying a sense as *O* means that, when the sense is used in a text or conversation, we do not expect it to express subjectivity and, if the phrase or sentence containing it is subjective, the subjectivity is due to something else [WM, p 3]."

The following subjective examples are given in

WM:

His **alarm** grew.
alarm, dismay, consternation – (fear resulting from the awareness of danger)
=> fear, fearfulness, fright – (an emotion experienced in anticipation of some specific pain or danger (usually accompanied by a desire to flee or fight))

What's the **catch**?
catch – (a hidden drawback; "it sounds good but what's the catch?")
=> drawback – (the quality of being a hindrance; "he pointed out all the drawbacks to my plan")

The following objective examples are given in WM:

The **alarm** went off.
alarm, warning device, alarm system – (a device that signals the occurrence of some undesirable event)
=> device – (an instrumentality invented for a particular purpose; "the device is small enough to wear on your wrist"; "a device intended to conserve water")

He sold his **catch** at the market.
catch, haul – (the quantity that was caught; "the catch was only 10 fish")
=> indefinite quantity – (an estimated quantity)

WM performed an agreement study and report that good agreement (κ=0.74) can be achieved between human annotators labeling the subjectivity of senses. For a similar task, (Su and Markert, 2008) also report good agreement.

3 Related Work

Many methods have been developed for automatically identifying subjective (*opinion*, *sentiment*, *attitude*, *affect-bearing*, etc.) words, e.g., (Turney, 2002; Riloff and Wiebe, 2003; Kim and Hovy, 2004; Taboada et al., 2006; Takamura et al., 2006).

Five groups have worked on subjectivity sense labeling. WM and Su and Markert (2008) (hereafter *SM*) assign *S*/*O* labels to senses, while Esuli and Sebastiani (hereafter *ES*) (2006a; 2007), Andreevskaia and Bergler (hereafter *AB*) (2006b; 2006a), and (Valitutti et al., 2004) assign polarity labels.

WM, SM, and ES have evaluated their systems against manually annotated word-sense data. WM's annotations are described above; SM's are similar. In the scheme ES use (Cerini et al., 2007), senses are assigned three scores, for positivity, negativity, and neutrality. There is no unambiguous mapping between the labels of WM/SM and ES, first because WM/SM use distinct classes and ES use numerical ratings, and second because WM/SM distinguish between objective senses on the one hand and neutral subjective senses on the other, while those are both neutral in the scheme used by ES.

WM use an unsupervised corpus-based approach, in which subjectivity labels are assigned to word senses based on a set of distributionally similar words in a corpus annotated with subjective expressions. SM explore methods that use existing resources that do not require manually annotated data; they also implement a supervised system for comparison, which we will call *SMsup*. The other three groups start with positive and negative seed sets and expand them by adding synonyms and antonyms, and traversing horizontal links in WordNet. AB, ES, and SMsup additionally use information contained in glosses; AB also use hyponyms; SMsup also uses relation and POS features. AB perform multiple runs of their system to assign fuzzy categories to senses. ES use a semi-supervised, multiple-classifier learning approach. In a later paper, (Esuli and Sebastiani, 2007), ES again use information in glosses, applying a random walk ranking algorithm to a graph in which synsets are linked if a member of the first synset appears in the gloss of the second.

Like ES and SMsup, we use machine learning, but with more diverse sources of knowledge. Further, several of our features are novel for the task. The LCS features (Section 6.1) detect subjectivity by measuring the similarity of a candidate word sense with a seed set. WM also use a similarity measure, but as a way to filter the output of a measure of distributional similarity (selecting words for a given word sense), not as we do to cumulatively calculate the subjectivity of a word sense. Another novel aspect of our similarity features is that they are particularized to domain, which greatly reduces calculation. The domain subjectivity LCS features (Section 6.2) are also novel for our task. So is augmenting seed sets with monosemous words, for greater coverage without requiring human intervention or sacrificing quality. Note that none of our features as we specifically define them has been used in previous work; combining them together, our approach outperforms previous approaches.

4 Lexicon and Annotations

We use the subjectivity lexicon of (Wiebe and Riloff, 2005)[1] both to create a subjective seed set and to create the experimental data sets. The lexicon is a list of words and phrases that have subjective uses, though only word entries are used in this paper (i.e., we do not address phrases at this point). Some entries are from manually developed resources, including the General Inquirer, while others were derived from corpora using automatic methods.

Through manual review and empirical testing on data, (Wiebe and Riloff, 2005) divided the clues into strong (*strongsubj*) and weak (*weaksubj*) subjectivity clues. *Strongsubj* clues have subjective meanings with high probability, and *weaksubj* clues have subjective meanings with lower probability.

To support our experiments, we annotated the senses[2] of polysemous nouns selected from the lexicon, using WM's annotation scheme described in Section 2. Due to time constraints, only some of the data was labeled through consensus labeling by two annotators; the rest was labeled by one annotator.

Overall, 2875 senses for 882 words were annotated. Even though all are senses of words from the subjectivity lexicon, only 1383 (48%) of the senses are subjective.

The words labeled *strongsubj* are in fact less ambiguous than those labeled *weaksubj* in our analysis, thus supporting the reliability classifications in the lexicon. 55% (1038/1924) of the senses of *strongsubj* words are subjective, while only 36% (345/951) of the senses of *weaksubj* words are subjective.

For the analysis in Section 7.3, we form subsets of the data annotated here to test performance of our method on different data compositions.

5 Seed Sets

Both subjective and objective seed sets are used to define the features described below. For seeds, a large number is desirable for greater coverage, although high quality is also important. We begin to build our subjective seed set by adding the monosemous *strongsubj* nouns of the subjectivity lexicon (there are 397 of these). Since they are monosemous, they pose no problem of sense ambiguity. We then expand the set with their hyponyms, as they were found useful in previous work by AB (2006b; 2006a). This yields a subjective seed set of 645 senses. After removing the word senses that belong to the same synset, so that only one word sense per synset is left, we ended up with 603 senses.

To create the objective seed set, two annotators manually annotated 800 random senses from WordNet, and selected for the objective seed set the ones they both agreed are clearly objective. This creates an objective seed set of 727. Again we removed multiple senses from the same synset leaving us with 722. The other 73 senses they annotated are added to the mixed data set described below. As this sampling shows, WordNet nouns are highly skewed toward objective senses, so finding an objective seed set is not difficult.

6 Features

6.1 Sense Subjectivity LCS Feature

This feature measures the similarity of a target sense with members of the subjective seed set. Here, similarity between two senses is determined by the extent to which they share information, measured by using the information content associated with their least common subsumer. For an intuition behind this feature, consider this example. In WordNet, the hypernym of the "strong criticism" sense of *attack* is *criticism*. Several other negative subjective senses are descendants of *criticism*, including the relevant senses of *fire*, *thrust*, and *rebuke*. Going up one more level, the hypernym of *criticism* is the "expression of disapproval" meaning of *disapproval*, which has several additional negative subjective descendants, such as the "expression of opposition and disapproval" sense of *discouragement*. Our hypothesis is that the cases where subjectivity is preserved in the hypernym structure, or where hypernyms do lead from subjective senses to others, *are* the ones that have the highest least common subsumer score with the seed set of known subjective senses.

We calculate similarity using the information-content based measure proposed in (Resnik, 1995), as implemented in the WordNet::Similarity package (using the default option in which LCS values are computed over the SemCor corpus).[3] Given a

[1] Available at http://www.cs.pitt.edu/mpqa

[2] In WordNet 2.0

[3] http://search.cpan.org/dist/WordNet-Similarity/

taxonomy such as WordNet, the information content associated with a concept is determined as the likelihood of encountering that concept, defined as $-log(p(C))$, where $p(C)$ is the probability of seeing concept C in a corpus. The similarity between two concepts is then defined in terms of information content as: $LCS_s(C_1, C_2) = max[-log(p(C))]$, where C is the concept that subsumes both C_1 and C_2 and has the highest information content (i.e., it is the *least common subsumer (LCS)*).

For this feature, a score is assigned to a target sense based on its semantic similarity to the members of a seed set; in particular, the maximum such similarity is used.

For a target sense t and a seed set S, we could have used the following score:

$$Score(t, S) = \max_{s \in S} LCS_s(t, s)$$

However, several researchers have noted that subjectivity may be domain specific. A version of WordNet exists, WordNet Domains (Gliozzo et al., 2005), which associates each synset with one of the domains in the Dewey Decimal library classification. After sorting our subjective seed set into different domains, we observed that over 80% of the subjective seed senses are concentrated in six domains (the rest are distributed among 35 domains).

Thus, we decided to particularize the semantic similarity feature to domain, such that only the subset of the seed set in the same domain as the target sense is used to compute the feature. This involves much less calculation, as LCS values are calculated only with respect to a subset of the seed set. We hypothesized that this would still be an effective feature, while being more efficient to calculate. This will be important when this method is applied to large resources such as the entire WordNet.

Thus, for seed set S and target sense t which is in domain D, the feature is defined as the following score:

$$SenseLCSscore(t, D, S) = \max_{d \in D \cap S} LCS_s(t, d)$$

The seed set is a parameter, so we could have defined a feature reflecting similarity to the objective seed set as well. Since WordNet is already highly skewed toward objective noun senses, any naive classifier need only guess the majority class for high accuracy for the objective senses. We included only a subjective feature to put more emphasis on the subjective senses. In the future, features could be defined with respect to objectivity, as well as polarity and other properties of subjectivity.

6.2 Domain Subjectivity LCS Score

We also include a feature reflecting the subjectivity of the domain of the target sense. Domains are assigned scores as follows. For domain D and seed set S:

$$DomainLCSscore(D, S) = ave_{d \in D \cap S} MemLCSscore(d, D, S)$$

where:

$$MemLCSscore(d, D, S) = \max_{d_i \in D \cap S, d_i \neq d} LCS_s(d, d_i)$$

The value of this feature for a sense is the score assigned to that sense's domain.

6.3 Common Related Senses

This feature is based on the intersection between the set of senses related (via WordNet relations) to the target sense and the set of senses related to members of a seed set. First, for the target sense and each member of the seed set, a set of related senses is formed consisting of its synonyms, antonyms and direct hypernyms as defined by WordNet. For a sense s, $R(s)$ is s together with its related senses.

Then, given a target sense t and a seed set S we compute an average percentage overlap as follows:

$$RelOverlap(t, S) = \frac{\sum_{s_i \in S} \frac{|R(t) \cap R(s_i)|}{\max(|R(t)|, |R(s_i)|)}}{|S|}$$

The value of a feature is its score. Two features are included in the experiments below, one for each of the subjective and objective seed sets.

6.4 Gloss-based features

These features are Lesk-style features (Lesk, 1986) that exploit overlaps between glosses of target and seed senses. We include two types in our work.

6.4.1 Average Percentage Gloss Overlap Features

For a sense s, $gloss(s)$ is the set of stems in the gloss of s (excluding stop words). Then, given a tar-

get sense t and a seed set S, we compute an average percentage overlap as follows:

$$GlOverlap(t,S) = \frac{\sum_{s_i \in S} \frac{\left|gloss(t) \cap \cup_{r \in R(s_i)} gloss(r)\right|}{\max(|gloss(t)|, |\cup_{r \in R(s_i)} gloss(r)|)}}{|S|}$$

As above, $R(s)$ is considered for each seed sense s, but now only the target sense t is considered, not $R(t)$. We did this because we hypothesized that the gloss can provide sufficient context for a given target sense, so that the addition of related words is not necessary.

We include two features, one for each of the subjective and objective seed sets.

6.4.2 Vector Gloss Overlap Features

For this feature we also consider overlaps of stems in glosses (excluding stop words). The overlaps considered are between the gloss of the target sense t and the glosses of $R(s)$ for all s in a seed set (for convenience, we will refer to these as $seedRelationSets$).

A vector of stems is created, one for each stem (excluding stop words) that appears in a gloss of a member of $seedRelationSets$. If a stem in the gloss of the target sense appears in this vector, then the vector entry for that stem is the total count of that stem in the glosses of the target sense and all members of $seedRelationSets$.

A feature is created for each vector entry whose value is the count at that position. Thus, these features consider counts of individual stems, rather than average proportions of overlaps, as for the previous type of gloss feature.

Two vectors of features are used, one where the seed set is the subjective seed set, and one where it is the objective seed set.

6.5 Summary

In summary, we use the following features (here, SS is the subjective seed set and OS is the objective one).

1. $SenseLCSscore(t, D, SS)$
2. $DomainLCSscore(D, SS)$
3. $RelOverlap(t, SS)$
4. $RelOverlap(t, OS)$
5. $GlOverlap(t, SS)$
6. $GlOverlap(t, OS)$

Features	Acc	P	R	F
All	77.3	72.8	74.3	73.5
Standalone Ablation Results				
All	77.3	72.8	74.3	73.5
LCS	68.2	69.3	44.2	54.0
Gloss vector	74.3	71.2	68.5	69.8
Overlaps	69.4	75.8	40.6	52.9
Leave-One-Out Ablation Results				
All	77.3	72.8	74.3	73.5
LCS	75.2	70.9	70.6	70.7
Gloss vector	75.0	74.4	61.8	67.5
Overlaps	74.8	71.9	73.8	72.8

Table 1: Results for the mixed corpus (2354 senses, 57.82% O))

7. *Vector of gloss words (SS)*
8. *Vector of gloss words (OS)*

7 Experiments

We perform 10-fold cross validation experiments on several data sets, using SVM_light (Joachims, 1999)[4] under its default settings.

Based on our random sampling of WordNet, it appears that WordNet nouns are highly skewed toward objective senses. (Esuli and Sebastiani, 2007) argue that random sampling from WordNet would yield a corpus mostly consisting of objective (neutral) senses, which would be "pretty useless as a benchmark for testing derived lexical resources for opinion mining [p. 428]." So, they use a mixture of subjective and objective senses in their data set.

To create a mixed corpus for our task, we annotated a second random sample from WordNet (which is as skewed as the previously mentioned one). We added together all of the senses of words in the lexicon which we annotated, the leftover senses from the selection of objective seed senses, and this new sample. We removed duplicates, multiple senses from the same synset, and any senses belonging to the same synset in either of the seed sets. This resulted in a corpus of 2354 senses, 993 (42.18%) of which are subjective and 1361 (57.82%) of which are objective.

The results with all of our features on this mixed corpus are given in Row 1 of Table 1. In Table 1, the

[4] http://svmlight.joachims.org/

first column identifies the features, which in this case is all of them. The next three columns show overall accuracy, and precision and recall for finding subjective senses. The baseline accuracy for the mixed data set (guessing the more frequent class, which is objective) is 57.82%. As the table shows, the accuracy is substantially above baseline.[5]

7.1 Analysis and Discussion

In this section, we seek to gain insights by performing ablation studies, evaluating our method on different data compositions, and comparing our results to previous results.

7.2 Ablation Studies

Since there are several features, we divided them into sets for the ablation studies. The vector-of-gloss-words features are the most similar to ones used in previous work. Thus, we opted to treat them as one ablation group (*Gloss vector*). The *Overlaps* group includes the $RelOverlap(t, SS)$, $RelOverlap(t, OS)$, $GlOverlap(t, SS)$, and $GlOverlap(t, OS)$ features. Finally, the *LCS* group includes the $SenseLCSscore$ and the $DomainLCSscore$ features.

There are two types of ablation studies. In the first, one group of features at a time is included. Those results are in the middle section of Table 1. Thus, for example, the row labeled *LCS* in this section is for an experiment using only the *LCS* features. In comparison to performance when all features are used, F-measure for the *Overlaps* and *LCS* ablations is significantly different at the $p < .01$ level, and, for the *Gloss Vector* ablation, it is significantly different at the $p = .052$ level (one-tailed t-test). Thus, all of the features together have better performance than any single type of feature alone.

In the second type of ablation study, we use all the features minus one group of features at a time. The results are in the bottom section of Table 1. Thus, for example, the row labeled *LCS* in this section is for an experiment using all but the *LCS* features. F-measures for *LCS* and *Gloss vector* are significantly different at the $p = .056$ and $p = .014$ levels, respectively. However, F-measure for the *Overlaps* ablation is not significantly different ($p = .39$).

[5]Note that, because the majority class is *O*, baseline recall (and thus F-measure) is 0.

Data (#senses)	Acc	P	R	F
mixed (2354 57.8% O)	77.3	72.8	74.3	73.5
strong+weak (1132)	77.7	76.8	78.9	77.8
weaksubj (566)	71.3	70.3	71.1	70.7
strongsubj (566)	78.6	78.8	78.6	78.7

Table 2: Results for different data sets (all are 50% S, unless otherwise notes)

These results provide evidence that *LCS* and *Gloss vector* are better together than either of them alone.

7.3 Results on Different Data Sets

Several methods have been developed for identifying subjective words. Perhaps an effective strategy would be to begin with a word-level subjectivity lexicon, and then perform subjectivity sense labeling to sort the subjective from objective senses of those words. We also wondered about the relative effectiveness of our method on *strongsubj* versus *weaksubj* clues.

To answer these questions, we apply the full model (again in 10-fold cross validation experiments) to data sets composed of senses of polysemous words in the subjectivity lexicon. To support comparison, all of the data sets in this section have a 50%-50% objective/subjective distribution.[6] The results are presented in Table 2.

For comparison, the first row repeats the results for the mixed corpus from Table 1. The second row shows results for a corpus of senses of a mixture of *strongsubj* and *weaksubj* words. The corpus was created by selecting a mixture of *strongsubj* and *weaksubj* words, extracting their senses and the *S*/*O* labels applied to them in Section 4, and then randomly removing senses of the more frequent class until the distribution is uniform. We see that the results on this corpus are better than on the mixed data set, even though the baseline accuracy is lower and the corpus is smaller. This supports the idea that an effective strategy would be to first identify opinion-bearing words, and then apply our method to those words to sort out their subjective and objective senses.

The third row shows results for a *weaksubj* subset

[6]As with the mixed data set, we removed from these data sets multiple senses from the same synset and any senses in the same synset in either of the seed sets.

Method	P	R	F
Our method	56.8	66.0	61.1
WM, 60% recall	44.0	66.0	52.8
SentiWordNet mapping	60.0	17.3	26.8

Table 3: Results for WM Corpus (212 senses, 76% O)

Method	A	P	R	F
Our Method	81.3%	60.3%	63.3%	61.8%
SM CV*	82.4%	70.8%	41.1%	52.0%
SM SL*	78.3%	53.0%	57.4%	54.9%

Table 4: Results for SM Corpus (484 senses, 76.9% O)

of the *strong+weak* corpus and the fourth shows results for a *strongsubj* subset that is of the same size. As expected, the results for the *weaksubj* senses are lower while those for the *strongsubj* senses are higher, as *weaksubj* clues are more ambiguous.

7.4 Comparisons with Previous Work

WM and SM address the same task as we do. To compare our results to theirs, we apply our full model (in 10-fold cross validation experiments) to their data sets.[7]

Table 3 has the WM data set results. WM rank their senses and present their results in the form of precision recall curves. The second row of Table 3 shows their results at the recall level achieved by our method (66%). Their precision at that level is substantially below ours.

Turning to ES, to create *S*/*O* annotations, we applied the following heuristic mapping (which is also used by SM for the purpose of comparison): any sense for which the sum of positive and negative scores is greater than or equal to 0.5 is S, otherwise it is O. We then evaluate the mapped tags against the gold standard of WM. The results are in Row 3 of Table 3. Note that this mapping is not fair to SentiWordNet, as the tasks are quite different, and we do not believe any conclusions can be drawn. We include the results to eliminate the possibility that their method is as good ours on our task, despite the differences between the tasks.

Table 4 has the results for the noun subset of SM's data set, which is the data set used by ES, reannotated by SM. CV* is their supervised system and SL* is their best non-supervised one. Our method has higher F-measure than the others.[8] Note that the focus of SM's work is not supervised machine learning.

8 Conclusions

In this paper, we introduced an integrative approach to automatic subjectivity word sense labeling which combines features exploiting the hierarchical structure and domain information of WordNet, as well as similarity of glosses and overlap among sets of semantically related words. There are several contributions. First, we learn several things. We found (in Section 4) that even reliable lists of subjective (opinion-bearing) words have many objective senses. We asked if word- and sense-level approaches could be used effectively in tandem, and found (in Section 7.3) that an effective strategy is to first identify opinion-bearing words, and then apply our method to sort out their subjective and objective senses. We also found (in Section 7.2) that the entire set of features gives better results than any individual type of feature alone.

Second, several of the features are novel for our task, including those exploiting the hierarchical structure of a lexical resource, domain information, and relations to seed sets expanded with monosemous senses.

Finally, the combination of our particular features is effective. For example, on senses of words from a subjectivity lexicon, accuracies range from 20 to 29 percentage points above baseline. Further, our combination of features outperforms previous approaches.

Acknowledgments

This work was supported in part by National Science Foundation awards #0840632 and #0840608. The authors are grateful to Fangzhong Su and Katja Markert for making their data set available, and to the three paper reviewers for their helpful suggestions.

[7] The WM data set is available at http://www.cs.pitt.edu/www.cs.pitt.edu/~wiebe. ES applied their method in (2006b) to WordNet, and made the results available as *SentiWordNet* at http://sentiwordnet.isti.cnr.it/.

[8] We performed the same type of evaluation as in SM's paper. That is, we assign a subjectivity label to one word sense for each synset, which is the same as applying a subjectivity label to a synset as a whole as done by SM.

References

Alina Andreevskaia and Sabine Bergler. 2006a. Mining wordnet for a fuzzy sentiment: Sentiment tag extraction from wordnet glosses. In *Proceedings of the 11rd Conference of the European Chapter of the Association for Computational Linguistics*.

Alina Andreevskaia and Sabine Bergler. 2006b. Sentiment tag extraction from wordnet glosses. In *Proceedings of 5th International Conference on Language Resources and Evaluation*.

Rebecca Bruce and Janyce Wiebe. 1999. Recognizing subjectivity: A case study of manual tagging. *Natural Language Engineering*, 5(2):187–205.

S. Cerini, V. Campagnoni, A. Demontis, M. Formentelli, and C. Gandini. 2007. Micro-wnop: A gold standard for the evaluation of automatically compiled lexical resources for opinion mining. In *Language resources and linguistic theory: Typology, second language acquisition, English linguistics*. Milano.

Andrea Esuli and Fabrizio Sebastiani. 2006a. Determining term subjectivity and term orientation for opinion mining. In *11th Meeting of the European Chapter of the Association for Computational Linguistics*.

Andrea Esuli and Fabrizio Sebastiani. 2006b. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation*, Genova, IT.

Andrea Esuli and Fabrizio Sebastiani. 2007. PageRanking wordnet synsets: An application to opinion mining. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 424–431, Prague, Czech Republic, June.

A. Gliozzo, C. Strapparava, E. d'Avanzo, and B. Magnini. 2005. Automatic acquisition of domain specific lexicons. Tech. report, IRST, Italy.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Scholkopf, C. Burgess, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, Cambridge, MA. MIT-Press.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the Twentieth International Conference on Computational Linguistics*, pages 1267–1373, Geneva, Switzerland.

Soo-Min Kim and Eduard Hovy. 2006. Identifying and analyzing judgment opinions. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 200–207, New York.

M.E. Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the SIGDOC Conference 1986*, Toronto, June.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* , pages 271–278, Barcelona, ES. Association for Computational Linguistics.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. International Joint Conference on Artificial Intelligence.*

E. Riloff and J. Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Conference on Empirical Methods in Natural Language Processing*, pages 105–112.

Fangzhong Su and Katja Markert. 2008. From word to sense: a case study of subjectivity recognition. In *Proceedings of the 22nd International Conference on Computational Linguistics*, Manchester.

M. Taboada, C. Anthony, and K. Voll. 2006. Methods for creating semantic orientation databases. In *Proceedings of 5th International Conference on Language Resources and Evaluation* .

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2006. Latent variable models for semantic orientations of phrases. In *Proceedings of the 11th Meeting of the European Chapter of the Association for Computational Linguistics* , Trento, Italy.

P. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, Philadelphia.

Alessandro Valitutti, Carlo Strapparava, and Oliviero Stock. 2004. Developing affective lexical resources. *PsychNology Journal*, 2(1):61–83.

J. Wiebe and R. Mihalcea. 2006. Word sense and subjectivity. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia.

Janyce Wiebe and Ellen Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of the 6th International Conference on Intelligent Text Processing and Computational Linguistics* , pages 486–497, Mexico City, Mexico.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing* , pages 347–354, Vancouver, Canada.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Conference on Empirical Methods in Natural Language Processing* , pages 129–136, Sapporo, Japan.

A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches

Eneko Agirre† **Enrique Alfonseca**‡ **Keith Hall**‡ **Jana Kravalova**‡§ **Marius Paşca**‡ **Aitor Soroa**†
† IXA NLP Group, University of the Basque Country
‡ Google Inc.
§ Institute of Formal and Applied Linguistics, Charles University in Prague
{e.agirre,a.soroa}@ehu.es {ealfonseca,kbhall,mars}@google.com
kravalova@ufal.mff.cuni.cz

Abstract

This paper presents and compares WordNet-based and distributional similarity approaches. The strengths and weaknesses of each approach regarding similarity and relatedness tasks are discussed, and a combination is presented. Each of our methods independently provide the best results in their class on the RG and WordSim353 datasets, and a supervised combination of them yields the best published results on all datasets. Finally, we pioneer cross-lingual similarity, showing that our methods are easily adapted for a cross-lingual task with minor losses.

1 Introduction

Measuring semantic similarity and relatedness between terms is an important problem in lexical semantics. It has applications in many natural language processing tasks, such as Textual Entailment, Word Sense Disambiguation or Information Extraction, and other related areas like Information Retrieval. The techniques used to solve this problem can be roughly classified into two main categories: those relying on pre-existing knowledge resources (thesauri, semantic networks, taxonomies or encyclopedias) (Alvarez and Lim, 2007; Yang and Powers, 2005; Hughes and Ramage, 2007) and those inducing distributional properties of words from corpora (Sahami and Heilman, 2006; Chen et al., 2006; Bollegala et al., 2007).

In this paper, we explore both families. For the first one we apply graph based algorithms to WordNet, and for the second we induce distributional similarities collected from a 1.6 Terabyte Web corpus. Previous work suggests that distributional similarities suffer from certain limitations, which make them less useful than knowledge resources for semantic similarity. For example, Lin (1998b) finds similar phrases like *captive-westerner* which made sense only in the context of the corpus used, and Budanitsky and Hirst (2006) highlight other problems that stem from the imbalance and sparseness of the corpora. Comparatively, the experiments in this paper demonstrate that distributional similarities can perform as well as the knowledge-based approaches, and a combination of the two can exceed the performance of results previously reported on the same datasets. An application to cross-lingual (CL) similarity identification is also described, with applications such as CL Information Retrieval or CL sponsored search. A discussion on the differences between learning similarity and relatedness scores is provided.

The paper is structured as follows. We first present the WordNet-based method, followed by the distributional methods. Section 4 is devoted to the evaluation and results on the monolingual and cross-lingual tasks. Section 5 presents some analysis, including learning curves for distributional methods, the use of distributional similarity to improve WordNet similarity, the contrast between similarity and relatedness, and the combination of methods. Section 6 presents related work, and finally, Section 7 draws the conclusions and mentions future work.

2 WordNet-based method

WordNet (Fellbaum, 1998) is a lexical database of English, which groups nouns, verbs, adjectives and adverbs into sets of synonyms (synsets), each expressing a distinct concept. Synsets are interlinked with conceptual-semantic and lexical relations, including hypernymy, meronymy, causality, etc.

Given a pair of words and a graph-based representation of WordNet, our method has basically two

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 19–27,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

steps: We first compute the personalized PageRank over WordNet separately for each of the words, producing a probability distribution over WordNet synsets. We then compare how similar these two discrete probability distributions are by encoding them as vectors and computing the cosine between the vectors.

We represent WordNet as a graph $G = (V, E)$ as follows: graph nodes represent WordNet concepts (synsets) and dictionary words; relations among synsets are represented by undirected edges; and dictionary words are linked to the synsets associated to them by directed edges.

For each word in the pair we first compute a personalized PageRank vector of graph G (Haveliwala, 2002). Basically, personalized PageRank is computed by modifying the random jump distribution vector in the traditional PageRank equation. In our case, we concentrate all probability mass in the target word.

Regarding PageRank implementation details, we chose a damping value of 0.85 and finish the calculation after 30 iterations. These are default values, and we did not optimize them. Our similarity method is similar, but simpler, to that used by (Hughes and Ramage, 2007), which report very good results on similarity datasets. More details of our algorithm can be found in (Agirre and Soroa, 2009). The algorithm and needed resouces are publicly available[1].

2.1 WordNet relations and versions

The WordNet versions that we use in this work are the Multilingual Central Repository or MCR (Atserias et al., 2004) (which includes English WordNet version 1.6 and wordnets for several other languages like Spanish, Italian, Catalan and Basque), and WordNet version 3.0[2]. We used all the relations in MCR (except cooccurrence relations and selectional preference relations) and in WordNet 3.0. Given the recent availability of the disambiguated gloss relations for WordNet 3.0[3], we also used a version which incorporates these relations. We will refer to the three versions as MCR16, WN30 and WN30g, respectively. Our choice was mainly motivated by the fact that MCR contains tightly aligned wordnets of several languages (see below).

2.2 Cross-linguality

MCR follows the EuroWordNet design (Vossen, 1998), which specifies an InterLingual Index (ILI) that links the concepts across wordnets of different languages. The wordnets for other languages in MCR use the English WordNet synset numbers as ILIs. This design allows a decoupling of the relations between concepts (which can be taken to be language independent) and the links from each content word to its corresponding concepts (which is language dependent).

As our WordNet-based method uses the graph of the concepts and relations, we can easily compute the similarity between words from different languages. For example, consider a English-Spanish pair like *car* – *coche*. Given that the Spanish WordNet is included in MCR we can use MCR as the common knowledge-base for the relations. We can then compute the personalized PageRank for each of *car* and *coche* on the same underlying graph, and then compare the similarity between both probability distributions.

As an alternative, we also tried to use publicly available mappings for wordnets (Daude et al., 2000)[4] in order to create a 3.0 version of the Spanish WordNet. The mapping was used to link Spanish variants to 3.0 synsets. We used the English WordNet 3.0, including glosses, to construct the graph. The two Spanish WordNet versions are referred to as MCR16 and WN30g.

3 Context-based methods

In this section, we describe the distributional methods used for calculating similarities between words, and profiting from the use of a large Web-based corpus.

This work is motivated by previous studies that make use of search engines in order to collect cooccurrence statistics between words. Turney (2001) uses the number of hits returned by a Web search engine to calculate the Pointwise Mutual Information (PMI) between terms, as an indicator of synonymy. Bollegala et al. (2007) calculate a number of popular relatedness metrics based on page counts,

[1] http://http://ixa2.si.ehu.es/ukb/

[2] Available from http://http://wordnet.princeton.edu/

[3] http://wordnet.princeton.edu/glosstag

[4] http://www.lsi.upc.es/∼nlp/tools/download-map.php.

like PMI, the Jaccard coefficient, the Simpson coefficient and the Dice coefficient, which are combined with lexico-syntactic patterns as model features. The model parameters are trained using Support Vector Machines (SVM) in order to later rank pairs of words. A different approach is the one taken by Sahami and Heilman (2006), who collect snippets from the results of a search engine and represent each snippet as a vector, weighted with the tf·idf score. The semantic similarity between two queries is calculated as the inner product between the centroids of the respective sets of vectors.

To calculate the similarity of two words w_1 and w_2, Ruiz-Casado et al. (2005) collect snippets containing w_1 from a Web search engine, extract a context around it, replace it with w_2 and check for the existence of that modified context in the Web.

Using a search engine to calculate similarities between words has the drawback that the data used will always be truncated. So, for example, the numbers of hits returned by search engines nowadays are always approximate and rounded up. The systems that rely on collecting snippets are also limited by the maximum number of documents returned per query, typically around a thousand. We hypothesize that by crawling a large corpus from the Web and doing standard corpus analysis to collect precise statistics for the terms we should improve over other unsupervised systems that are based on search engine results, and should yield results that are competitive even when compared to knowledge-based approaches.

In order to calculate the semantic similarity between the words in a set, we have used a vector space model, with the following three variations:

In the **bag-of-words approach**, for each word w in the dataset we collect every term t that appears in a window centered in w, and add them to the vector together with its frequency.

In the **context window approach**, for each word w in the dataset we collect every window W centered in w (removing the central word), and add it to the vector together with its frequency (the total number of times we saw window W around w in the whole corpus). In this case, all punctuation symbols are replaced with a special token, to unify patterns like *, the <term> said to* and *' the <term> said to*. Throughout the paper, when we mention a context window of size N it means N words at each side of the phrase of interest.

In the **syntactic dependency approach**, we parse the entire corpus using an implementation of an Inductive Dependency parser as described in Nivre (2006). For each word w we collect a template of the syntactic context. We consider sequences of governing words (e.g. the parent, grand-parent, etc.) as well as collections of descendants (e.g., immediate children, grandchildren, etc.). This information is then encoded as a contextual template. For example, the context template *cooks <term> delicious* could be contexts for nouns such as *food, meals, pasta*, etc. This captures both syntactic preferences as well as selectional preferences. Contrary to Pado and Lapata (2007), we do not use the labels of the syntactic dependencies.

Once the vectors have been obtained, the frequency for each dimension in every vector is weighted using the other vectors as contrast set, with the χ^2 test, and finally the cosine similarity between vectors is used to calculate the similarity between each pair of terms.

Except for the syntactic dependency approach, where closed-class words are needed by the parser, in the other cases we have removed stopwords (pronouns, prepositions, determiners and modal and auxiliary verbs).

3.1 Corpus used

We have used a corpus of four billion documents, crawled from the Web in August 2008. An HTML parser is used to extract text, the language of each document is identified, and non-English documents are discarded. The final corpus remaining at the end of this process contains roughly 1.6 Terawords. All calculations are done in parallel sharding by dimension, and it is possible to calculate all pairwise similarities of the words in the test sets very quickly on this corpus using the MapReduce infrastructure. A complete run takes around 15 minutes on 2,000 cores.

3.2 Cross-linguality

In order to calculate similarities in a cross-lingual setting, where some of the words are in a language l other than English, the following algorithm is used:

Method	Window size	RG dataset	WordSim353 dataset
MCR16		0.83 [0.73, 0.89]	0.53 (0.56) [0.45, 0.60]
WN30		0.79 [0.67, 0.86]	0.56 (0.58) [0.48, 0.63]
WN30g		0.83 [0.73, 0.89]	**0.66 (0.69)** [0.59, 0.71]
CW	1	0.83 [0.73, 0.89]	0.63 [0.57, 0.69]
	2	0.83 [0.74, 0.90]	0.60 [0.53, 0.66]
	3	0.85 [0.76, 0.91]	0.59 [0.52, 0.65]
	4	**0.89** [0.82, 0.93]	0.60 [0.53, 0.66]
	5	0.80 [0.70, 0.88]	0.58 [0.51, 0.65]
	6	0.75 [0.62, 0.84]	0.58 [0.50, 0.64]
	7	0.72 [0.58, 0.82]	0.57 [0.49, 0.63]
BoW	1	0.81 [0.70, 0.88]	0.64 [0.57, 0.70]
	2	0.80 [0.69, 0.87]	0.64 [0.58, 0.70]
	3	0.79 [0.67, 0.86]	0.64 [0.58, 0.70]
	4	0.78 [0.66, 0.86]	0.65 [0.58, 0.70]
	5	0.77 [0.64, 0.85]	0.64 [0.58, 0.70]
	6	0.76 [0.63, 0.85]	0.65 [0.58, 0.70]
	7	0.75 [0.62, 0.84]	0.64 [0.58, 0.70]
Syn	G1,D0	0.81 [0.70, 0.88]	0.62 [0.55, 0.68]
	G2,D0	0.82 [0.72, 0.89]	0.55 [0.48, 0.62]
	G3,D0	0.81 [0.71, 0.88]	0.62 [0.56, 0.68]
	G1,D1	0.82 [0.72, 0.89]	0.62 [0.55, 0.68]
	G2,D1	0.82 [0.73, 0.89]	0.62 [0.55, 0.68]
	G3,D1	0.82 [0.72, 0.88]	0.62 [0.55, 0.68]
CW+ Syn	4; G1,D0	0.88 [0.81, 0.93]	**0.66** [0.59, 0.71]
	4; G2,D0	0.87 [0.80, 0.92]	0.64 [0.57, 0.70]
	4; G3,D0	0.86 [0.77, 0.91]	0.63 [0.56, 0.69]
	4; G1,D1	0.83 [0.73, 0.89]	0.48 [0.40, 0.56]
	4; G2,D1	0.83 [0.73, 0.89]	0.49 [0.40, 0.56]
	4; G3,D1	0.82 [0.72, 0.89]	0.48 [0.40, 0.56]

Table 1: Spearman correlation results for the various WordNet-based models and distributional models. CW=Context Windows, BoW=bag of words, Syn=syntactic vectors. For Syn, the *window size* is actually the tree-depth for the governors and descendants. For examples, G1 indicates that the contexts include the parents and D2 indicates that both the children and grandchildren make up the contexts. The final grouping includes both contextual windows (at width 4) and syntactic contexts in the template vectors. Max scores are bolded.

1. Replace each non-English word in the dataset with its 5-best translations into English using state-of-the-art machine translation technology.
2. The vector corresponding to each Spanish word is calculated by collecting features from all the contexts of any of its translations.
3. Once the vectors are generated, the similarities are calculated in the same way as before.

4 Experimental results

4.1 Gold-standard datasets

We have used two standard datasets. The first one, RG, consists of 65 pairs of words collected by Rubenstein and Goodenough (1965), who had them judged by 51 human subjects in a scale from 0.0 to 4.0 according to their similarity, but ignoring any other possible semantic relationships that might appear between the terms. The second dataset, WordSim353[5] (Finkelstein et al., 2002) contains 353 word pairs, each associated with an average of 13 to 16 human judgements. In this case, both similarity and relatedness are annotated without any distinction. Several studies indicate that the human scores consistently have very high correlations with each other (Miller and Charles, 1991; Resnik, 1995), thus validating the use of these datasets for evaluating semantic similarity.

[5]Available at http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/wordsim353.html

Context	RG terms and frequencies
ll never forget the * on his face when	grin,2,smile,10
he had a giant * on his face and	grin,3,smile,2
room with a huge * on her face and	grin,2,smile,6
the state of every * will be updated every	automobile,2,car,3
repair or replace the * if it is stolen	automobile,2,car,2
located on the north * of the Bay of	shore,14,coast,2
areas on the eastern * of the Adriatic Sea	shore,3,coast,2
Thesaurus of Current English * The Oxford Pocket Thesaurus	slave,3,boy,5,shore,3,string,2
	wizard,4,glass,4,crane,5,smile,5
	implement,5,oracle,2,lad,2
	food,3,car,2,madhouse,3,jewel,3
	asylum,4,tool,8,journey,6,*etc.*
be understood that the * 10 may be designed	crane,3,tool,3
a fight between a * and a snake and	bird,3,crane,5

Table 2: Sample of context windows for the terms in the RG dataset.

For the cross-lingual evaluation, the two datasets were modified by translating the second word in each pair into Spanish. Two humans translated simultaneously both datasets, with an inter-tagger agreement of 72% for RG and 84% for WordSim353.

4.2 Results

Table 1 shows the Spearman correlation obtained on the RG and WordSim353 datasets, including the interval at 0.95 of confidence[6].

Overall the distributional context-window approach performs best in the RG, reaching 0.89 correlation, and both WN30g and the combination of context windows and syntactic context perform best on WordSim353. Note that the confidence intervals are quite large in both RG and WordSim353, and few of the pairwise differences are statistically significant.

Regarding WordNet-based approaches, the use of the glosses and WordNet 3.0 (WN30g) yields the best results in both datasets. While MCR16 is close to WN30g for the RG dataset, it lags well behind on WordSim353. This discrepancy is further analyzed is Section 5.3. Note that the performance of WordNet in the WordSim353 dataset suffers from unknown words. In fact, there are nine pairs which returned null similarity for this reason. The num-

[6]To calculate the Spearman correlations values are transformed into ranks, and we calculate the Pearson correlation on them. The confidence intervals refer to the Pearson correlations of the rank vectors.

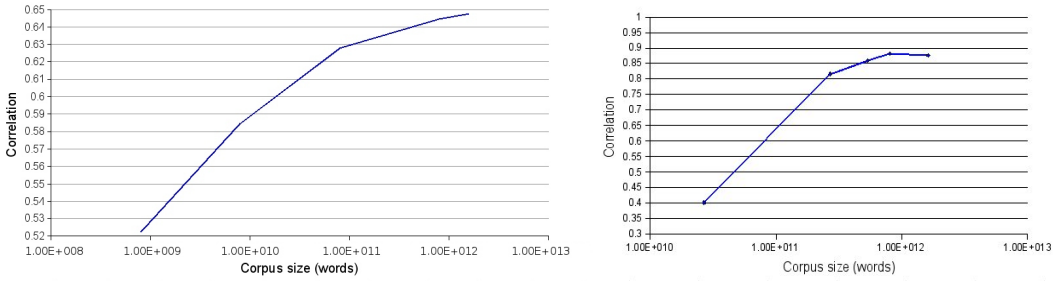


Figure 1: Effect of the size of the training corpus, for the best distributional similarity model in each dataset. Left: WordSim353 with bag-of-words, Right: RG with context windows.

Dataset	Method	overall	Δ	interval
RG	MCR16	0.78	-0.05	[0.66, 0.86]
	WN30g	0.74	-0.09	[0.61, 0.84]
	Bag of words	0.68	-0.23	[0.53, 0.79]
	Context windows	**0.83**	-0.05	[0.73, 0.89]
WS353	MCR16	0.42 (0.53)	-0.11 (-0.03)	[0.34, 0.51]
	WN30g	**0.58** (0.67)	-0.07 (-0.02)	[0.51, 0.64]
	Bag of words	0.53	-0.12	[0.45, 0.61]
	Context windows	0.52	-0.11	[0.44, 0.59]

Table 3: Results obtained by the different methods on the Spanish/English cross-lingual datasets. The Δ column shows the performance difference with respect to the results on the original dataset.

ber in parenthesis in Table 1 for WordSim353 shows the results for the 344 remaining pairs. Section 5.2 shows a proposal to overcome this limitation.

The bag-of-words approach tends to group together terms that can have a similar distribution of contextual terms. Therefore, terms that are topically related can appear in the same textual passages and will get high values using this model. We see this as an explanation why this model performed better than the context window approach for WordSim353, where annotators were instructed to provide high ratings to related terms. On the contrary, the context window approach tends to group together words that are exchangeable in exactly the same context, preserving order. Table 2 illustrates a few examples of context collected. Therefore, true synonyms and hyponyms/hyperonyms will receive high similarities, whereas terms related topically or based on any other semantic relation (e.g. *movie* and *star*) will have lower scores. This explains why this method performed better for the RG dataset. Section 5.3 confirms these observations.

4.3 Cross-lingual similarity

Table 3 shows the results for the English-Spanish cross-lingual datasets. For RG, MCR16 and the context windows methods drop only 5 percentage points, showing that cross-lingual similarity is feasible, and that both cross-lingual strategies are robust.

The results for WordSim353 show that WN30g is the best for this dataset, with the rest of the methods falling over 10 percentage points relative to the monolingual experiment. A closer look at the WordNet results showed that most of the drop in performance was caused by out-of-vocabulary words, due to the smaller vocabulary of the Spanish WordNet. Though not totally comparable, if we compute the correlation over pairs covered in WordNet alone, the correlation would drop only 2 percentage points. In the case of the distributional approaches, the fall in performance was caused by the translations, as only 61% of the words were translated into the original word in the English datasets.

5 Detailed analysis and system combination

In this section we present some analysis, including learning curves for distributional methods, the use of distributional similarity to improve WordNet similarity, the contrast between similarity and relatedness, and the combination of methods.

5.1 Learning curves for distributional methods

Figure 1 shows that the correlation improves with the size of the corpus, as expected. For the results using the WordSim353 corpus, we show the results of the bag-of-words approach with context size 10. Results improve from 0.5 Spearman correlation up to 0.65 when increasing the corpus size three orders of magnitude, although the effect decays at the end, which indicates that we might not get fur-

Method	Without similar words	With similar words
WN30	0.56 (0.58) [0.48, 0.63]	**0.58** [0.51, 0.65]
WN30g	0.66 (0.69) [0.59, 0.71]	**0.68** [0.62, 0.73]

Table 4: Results obtained replacing unknown words with their most similar three words (WordSim353 dataset).

Method	overall	Similarity	Relatedness
MCR16	0.53 [0.45, 0.60]	0.65 [0.56, 0.72]	0.33 [0.21, 0.43]
WN30	0.56 [0.48, 0.63]	0.73 [0.65, 0.79]	0.38 [0.27, 0.48]
WN30g	**0.66** [0.59, 0.71]	0.72 [0.64, 0.78]	0.56 [0.46, 0.64]
BoW	0.65 [0.59, 0.71]	0.70 [0.63, 0.77]	**0.62** [0.53, 0.69]
CW	0.60 [0.53, 0.66]	**0.77** [0.71, 0.82]	0.46 [0.36, 0.55]

Table 5: Results obtained on the WordSim353 dataset and on the two similarity and relatedness subsets.

ther gains going beyond the current size of the corpus. With respect to results for the RG dataset, we used a context-window approach with context radius 4. Here, results improve even more with data size, probably due to the sparse data problem collecting 8-word context windows if the corpus is not large enough. Correlation improves linearly right to the end, where results stabilize around 0.89.

5.2 Combining both approaches: dealing with unknown words in WordNet

Although the vocabulary of WordNet is very extensive, applications are bound to need the similarity between words which are not included in WordNet. This is exemplified in the WordSim353 dataset, where 9 pairs contain words which are unknown to WordNet. In order to overcome this shortcoming, we could use similar words instead, as provided by the distributional thesaurus. We used the distributional thesaurus defined in Section 3, using context windows of width 4, to provide three similar words for each of the unknown words in WordNet. Results improve for both WN30 and WN30g, as shown in Table 4, attaining our best results for WordSim353.

5.3 Similarity vs. relatedness

We mentioned above that the annotation guidelines of WordSim353 did not distinguish between similar and related pairs. As the results in Section 4 show, different techniques are more appropriate to calculate either similarity or relatedness. In order to study this effect, ideally, we would have two versions of the dataset, where annotators were given precise instructions to distinguish similarity in one case, and relatedness in the other. Given the lack of such datasets, we devised a simpler approach in order to reuse the existing human judgements. We manually split the dataset in two parts, as follows.

First, two humans classified all pairs as being synonyms of each other, antonyms, identical, hyperonym-hyponym, hyponym-hyperonym, holonym-meronym, meronym-holonym, and none-of-the-above. The inter-tagger agreement rate was 0.80, with a Kappa score of 0.77. This annotation was used to group the pairs in three categories: similar pairs (those classified as synonyms, antonyms, identical, or hyponym-hyperonym), related pairs (those classified as meronym-holonym, and pairs classified as none-of-the-above, with a human average similarity greater than 5), and unrelated pairs (those classified as none-of-the-above that had average similarity less than or equal to 5). We then created two new gold-standard datasets: **similarity** (the union of similar and unrelated pairs), and **relatedness** (the union of related and unrelated)[7].

Table 5 shows the results on the relatedness and similarity subsets of WordSim353 for the different methods. Regarding WordNet methods, both WN30 and WN30g perform similarly on the similarity subset, but WN30g obtains the best results by far on the relatedness data. These results are congruent with our expectations: two words are similar if their synsets are in close places in the WordNet hierarchy, and two words are related if there is a connection between them. Most of the relations in WordNet are of hierarchical nature, and although other relations exist, they are far less numerous, thus explaining the good results for both WN30 and WN30g on similarity, but the bad results of WN30 on relatedness. The disambiguated glosses help find connections among related concepts, and allow our method to better model relatedness with respect to WN30.

The low results for MCR16 also deserve some comments. Given the fact that MCR16 performed very well on the RG dataset, it comes as a surprise that it performs so poorly for the similarity subset of WordSim353. In an additional evaluation, we attested that MCR16 does indeed perform as well as MCR30g on the **similar pairs** subset. We believe that this deviation could be due to the method used to construct the similarity dataset, which includes some pairs of loosely related pairs labeled as unrelated.

[7] Available at http://alfonseca.org/eng/research/wordsim353.html

Methods combined in the SVM	RG dataset	WordSim353 dataset	WordSim353 similarity	WordSim353 relatedness
WN30g, bag of words	0.88 [0.82, 0.93]	**0.78** [0.73, 0.81]	0.81 [0.76, 0.86]	**0.72** [0.65, 0.77]
WN30g, context windows	0.90 [0.84, 0.94]	0.73 [0.68, 0.79]	**0.83** [0.78, 0.87]	0.64 [0.56, 0.71]
WN30g, syntax	0.89 [0.83, 0.93]	0.75 [0.70, 0.79]	**0.83** [0.78, 0.87]	0.67 [0.60, 0.74]
WN30g, bag of words, context windows, syntax	**0.96** [0.93, 0.97]	**0.78** [0.73, 0.82]	**0.83** [0.78, 0.87]	0.71 [0.65, 0.77]

Table 6: Results using a supervised combination of several systems. Max values are bolded for each dataset.

Concerning the techniques based on distributional similarities, the method based on context windows provides the best results for similarity, and the bag-of-words representation outperforms most of the other techniques for relatedness.

5.4 Supervised combination

In order to gain an insight on which would be the upper bound that we could obtain when combining our methods, we took the output of three systems (bag of words with window size 10, context window with size 4, and the WN30g run). Each of these outputs is a ranking of word pairs, and we implemented an oracle that chooses, for each pair, the rank that is most similar to the rank of the pair in the gold-standard. The outputs of the oracle have a Spearman correlation of 0.97 for RG and 0.92 for WordSim353, which gives as an indication of the correlations that could be achieved by choosing for each pair the rank output by the best classifier for that pair.

The previous results motivated the use of a supervised approach to combine the output of the different systems. We created a training corpus containing pairs of pairs of words from the datasets, having as features the similarity and rank of each pair involved as given by the different unsupervised systems. A classifier is trained to decide whether the first pair is more similar than the second one. For example, a training instance using two unsupervised classifiers is

0.001364, 31, 0.327515, 64, 0.084805, 57, 0.109061, 59, negative

meaning that the similarities given by the first classifier to the two pairs were 0.001364 and 0.327515 respectively, which ranked them in positions 31 and 64. The second classifier gave them similarities of 0.084805 and 0.109061 respectively, which ranked them in positions 57 and 59. The class *negative* indicates that in the gold-standard the first pair has a lower score than the second pair.

We have trained a SVM to classify pairs of pairs, and use its output to rank the entries in both datasets. It uses a polynomial kernel with degree 4. We did not have a held-out set, so we used the standard settings of Weka, without trying to modify parameters, e.g. C. Each word pair is scored with the number of pairs that were considered to have less similarity using the SVM. The results using 10-fold cross-validation are shown in Table 6. A combination of all methods produces the best results reported so far for both datasets, statistically significant for RG.

Method	Source	Spearman (MC)	Pearson (MC)
(Sahami et al., 2006)	Web snippets	0.62 [0.32, 0.81]	0.58 [0.26, 0.78]
(Chen et al., 2006)	Web snippets	0.69 [0.42, 0.84]	0.69 [0.42, 0.85]
(Wu and Palmer, 1994)	WordNet	0.78 [0.59, 0.90]	0.78 [0.57, 0.89]
(Leacock et al., 1998)	WordNet	0.79 [0.59, 0.90]	0.82 [0.64, 0.91]
(Resnik, 1995)	WordNet	0.81 [0.62, 0.91]	0.80 [0.60, 0.90]
(Lin, 1998a)	WordNet	0.82 [0.65, 0.91]	0.83 [0.67, 0.92]
(Bollegala et al., 2007)	Web snippets	0.82 [0.64, 0.91]	0.83 [0.67, 0.92]
(Jiang and Conrath, 1997)	WordNet	0.83 [0.67, 0.92]	0.85 [0.69, 0.93]
(Jarmasz, 2003)	Roget's	0.87 [0.73, 0.94]	0.87 [0.74, 0.94]
(Patwardhan et al., 2006)	WordNet	n/a	0.91
(Alvarez and Lim, 2007)	WordNet	n/a	0.91
(Yang and Powers, 2005)	WordNet	0.87 [0.73, 0.91]	0.92 [0.84, 0.96]
(Hughes et al., 2007)	WordNet	0.90	n/a
Personalized PageRank	WordNet	0.89 [0.77, 0.94]	n/a
Bag of words	Web corpus	0.85 [0.70, 0.93]	0.84 [0.69, 0.93]
Context window	Web corpus	0.88 [0.76, 0.95]	0.89 [0.77, 0.95]
Syntactic contexts	Web corpus	0.76 [0.54, 0.88]	0.74 [0.51, 0.87]
SVM	Web, WN	**0.92** [0.84, 0.96]	**0.93** [0.85, 0.97]

Table 7: Comparison with previous approaches for MC.

6 Related work

Contrary to the WordSim353 dataset, common practice with the RG dataset has been to perform the evaluation with Pearson correlation. In our believe Pearson is less informative, as the Pearson correlation suffers much when the scores of two systems are not linearly correlated, something which happens often given due to the different nature of the techniques applied. Some authors, e.g. Alvarez and Lim (2007), use a non-linear function to map the system outputs into new values distributed more similarly to the values in the gold-standard. In their case, the mapping function was $\exp\left(\frac{-x}{4}\right)$, which was chosen empirically. Finding such a function is dependent on the dataset used, and involves an extra step in the similarity calculations. Alternatively, the Spearman correlation provides an evaluation metric that is independent of such data-dependent transformations.

Most similarity researchers have published their

Word pair	M&C	SVM	Word pair	M&C	SVM
automobile, car	3.92	62	crane, implement	1.68	26
journey, voyage	3.84	54	brother, lad	1.66	39
gem, jewel	3.84	61	car, journey	1.16	37
boy, lad	3.76	57	monk, oracle	1.1	32
coast, shore	3.7	53	food, rooster	0.89	3
asylum, madhouse	3.61	45	coast, hill	0.87	34
magician, wizard	3.5	49	forest, graveyard	0.84	27
midday, noon	3.42	61	monk, slave	0.55	17
furnace, stove	3.11	50	lad, wizard	0.42	13
food, fruit	3.08	47	coast, forest	0.42	18
bird, cock	3.05	46	cord, smile	0.13	5
bird, crane	2.97	38	glass, magician	0.11	10
implement, tool	2.95	55	rooster, voyage	0.08	1
brother, monk	2.82	42	noon, string	0.08	5

Table 8: Our best results for the MC dataset.

Method	Source	Spearman
(Strube and Ponzetto, 2006)	Wikipedia	0.19–0.48
(Jarmasz, 2003)	WordNet	0.33–0.35
(Jarmasz, 2003)	Roget's	0.55
(Hughes and Ramage, 2007)	WordNet	0.55
(Finkelstein et al., 2002)	Web corpus, WN	0.56
(Gabrilovich and Markovitch, 2007)	ODP	0.65
(Gabrilovich and Markovitch, 2007)	Wikipedia	0.75
SVM	Web corpus, WN	**0.78**

Table 9: Comparison with previous work for WordSim353.

complete results on a smaller subset of the RG dataset containing 30 word pairs (Miller and Charles, 1991), usually referred to as MC, making it possible to compare different systems using different correlation. Table 7 shows the results of related work on MC that was available to us, including our own. For the authors that did not provide the detailed data we include only the Pearson correlation with no confidence intervals.

Among the unsupervised methods introduced in this paper, the context window produced the best reported Spearman correlation, although the 0.95 confidence intervals are too large to allow us to accept the hypothesis that it is better than all others methods. The supervised combination produces the best results reported so far. For the benefit of future research, our results for the MC subset are displayed in Table 8.

Comparison on the WordSim353 dataset is easier, as all researchers have used Spearman. The figures in Table 9) show that our WordNet-based method outperforms all previously published WordNet methods. We want to note that our WordNet-based method outperforms that of Hughes and Ramage (2007), which uses a similar method. Although there are some differences in the method, we think that the main performance gain comes from the use of the disambiguated glosses, which they did not use. Our distributional methods also outperform all other corpus-based methods. The most similar approach to our distributional technique is Finkelstein et al. (2002), who combined distributional similarities from Web documents with a similarity from WordNet. Their results are probably worse due to the smaller data size (they used 270,000 documents) and the differences in the calculation of the similarities. The only method which outperforms our non-supervised methods is that of (Gabrilovich and Markovitch, 2007) when based on Wikipedia, probably because of the dense, manually distilled knowledge contained in Wikipedia. All in all, our supervised combination gets the best published results on this dataset.

7 Conclusions and future work

This paper has presented two state-of-the-art distributional and WordNet-based similarity measures, with a study of several parameters, including performance on similarity and relatedness data. We show that the use of disambiguated glosses allows for the best published results for WordNet-based systems on the WordSim353 dataset, mainly due to the better modeling of relatedness (as opposed to similarity). Distributional similarities have proven to be competitive when compared to knowledge-based methods, with context windows being better for similarity and bag of words for relatedness. Distributional similarity was effectively used to cover out-of-vocabulary items in the WordNet-based measure providing our best unsupervised results. The complementarity of our methods was exploited by a supervised learner, producing the best results so far for RG and WordSim353. Our results include confidence values, which, surprisingly, were not included in most previous work, and show that many results over RG and WordSim353 are indistinguishable. The algorithm for WordNet-base similarity and the necessary resources are publicly available[8].

This work pioneers cross-lingual extension and evaluation of both distributional and WordNet-based measures. We have shown that closely aligned wordnets provide a natural and effective way to compute cross-lingual similarity with minor losses. A simple translation strategy also yields good results for distributional methods.

[8] http://ixa2.si.ehu.es/ukb/

References

E. Agirre and A. Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proc. of EACL 2009*, Athens, Greece.

M.A. Alvarez and S.J. Lim. 2007. A Graph Modeling of Semantic Similarity between Words. *Proc. of the Conference on Semantic Computing*, pages 355–362.

J. Atserias, L. Villarejo, G. Rigau, E. Agirre, J. Carroll, B. Magnini, and P. Vossen. 2004. The meaning multilingual central repository. In *Proc. of Global WordNet Conference*, Brno, Czech Republic.

D. Bollegala, Matsuo Y., and M. Ishizuka. 2007. Measuring semantic similarity between words using web search engines. In *Proceedings of WWW'2007*.

A. Budanitsky and G. Hirst. 2006. Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Computational Linguistics*, 32(1):13–47.

H. Chen, M. Lin, and Y. Wei. 2006. Novel association measures using web search with double checking. In *Proceedings of COCLING/ACL 2006*.

J. Daude, L. Padro, and G. Rigau. 2000. Mapping WordNets using structural information. In *Proceedings of ACL'2000*, Hong Kong.

C. Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database and Some of its Applications*. MIT Press, Cambridge, Mass.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. 2002. Placing Search in Context: The Concept Revisited. *ACM Transactions on Information Systems*, 20(1):116–131.

E. Gabrilovich and S. Markovitch. 2007. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. *Proc of IJCAI*, pages 6–12.

T. H. Haveliwala. 2002. Topic-sensitive pagerank. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 517–526.

T. Hughes and D. Ramage. 2007. Lexical semantic relatedness with random graph walks. In *Proceedings of EMNLP-CoNLL-2007*, pages 581–589.

M. Jarmasz. 2003. Roget's Thesuarus as a lexical resource for Natural Language Processing.

J.J. Jiang and D.W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference on Research in Computational Linguistics*, volume 33. Taiwan.

C. Leacock and M. Chodorow. 1998. Combining local context and WordNet similarity for word sense identification. *WordNet: An Electronic Lexical Database*, 49(2):265–283.

D. Lin. 1998a. An information-theoretic definition of similarity. In *Proc. of ICML*, pages 296–304, Wisconsin, USA.

D. Lin. 1998b. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of ACL-98*.

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

J. Nivre. 2006. *Inductive Dependency Parsing*, volume 34 of *Text, Speech and Language Technology*. Springer.

S. Pado and M. Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

S. Patwardhan and T. Pedersen. 2006. Using WordNet-based Context Vectors to Estimate the Semantic Relatedness of Concepts. In *Proceedings of the EACL Workshop on Making Sense of Sense: Bringing Computational Linguistics and Pycholinguistics Together*, pages 1–8, Trento, Italy.

P. Resnik. 1995. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. *Proc. of IJCAI*, 14:448–453.

H. Rubenstein and J.B. Goodenough. 1965. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633.

M Ruiz-Casado, E. Alfonseca, and P. Castells. 2005. Using context-window overlapping in Synonym Discovery and Ontology Extension. In *Proceedings of RANLP-2005*, Borovets, Bulgaria,.

M. Sahami and T.D. Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. *Proc. of WWW*, pages 377–386.

M. Strube and S.P. Ponzetto. 2006. WikiRelate! Computing Semantic Relatedness Using Wikipedia. In *Proceedings of the AAAI-2006*, pages 1419–1424.

P.D. Turney. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. *Lecture Notes in Computer Science*, 2167:491–502.

P. Vossen, editor. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers.

Z. Wu and M. Palmer. 1994. Verb semantics and lexical selection. In *Proc. of ACL*, pages 133–138, Las Cruces, New Mexico.

D. Yang and D.M.W. Powers. 2005. Measuring semantic similarity in the taxonomy of WordNet. *Proceedings of the Australasian conference on Computer Science*.

A Fully Unsupervised Word Sense Disambiguation Method Using Dependency Knowledge

Ping Chen
Dept. of Computer and Math. Sciences
University of Houston-Downtown
chenp@uhd.edu

Wei Ding
Department of Computer Science
University of Massachusetts-Boston
ding@cs.umb.edu

Chris Bowes
Dept. of Computer and Math. Sciences
University of Houston-Downtown
bowesc@uhd.edu

David Brown
Dept. of Computer and Math. Sciences
University of Houston-Downtown
brownd@uhd.edu

Abstract

Word sense disambiguation is the process of determining which sense of a word is used in a given context. Due to its importance in understanding semantics of natural languages, word sense disambiguation has been extensively studied in Computational Linguistics. However, existing methods either are brittle and narrowly focus on specific topics or words, or provide only mediocre performance in real-world settings. Broad coverage and disambiguation quality are critical for a word sense disambiguation system. In this paper we present a fully unsupervised word sense disambiguation method that requires only a dictionary and unannotated text as input. Such an automatic approach overcomes the problem of brittleness suffered in many existing methods and makes broad-coverage word sense disambiguation feasible in practice. We evaluated our approach using SemEval 2007 Task 7 (Coarse-grained English All-words Task), and our system significantly outperformed the best unsupervised system participating in SemEval 2007 and achieved the performance approaching top-performing supervised systems. Although our method was only tested with coarse-grained sense disambiguation, it can be directly applied to fine-grained sense disambiguation.

1 Introduction

In many natural languages, a word can represent multiple meanings/senses, and such a word is called a homograph. Word sense disambiguation(WSD) is the process of determining which sense of a homograph is used in a given context. WSD is a long-standing problem in Computational Linguistics, and has significant impact in many real-world applications including machine translation, information extraction, and information retrieval. Generally, WSD methods use the context of a word for its sense disambiguation, and the context information can come from either annotated/unannotated text or other knowledge resources, such as WordNet (Fellbaum, 1998), SemCor (SemCor, 2008), Open Mind Word Expert (Chklovski and Mihalcea, 2002), eXtended WordNet (Moldovan and Rus, 2001), Wikipedia (Mihalcea, 2007), parallel corpora (Ng, Wang, and Chan, 2003). In (Ide and Véronis, 1998) many different WSD approaches were described. Usually, WSD techniques can be divided into four categories (Agirre and Edmonds, 2006),

- Dictionary and knowledge based methods. These methods use lexical knowledge bases such as dictionaries and thesauri, and hypothesize that context knowledge can be extracted from definitions of words. For example, Lesk disambiguated two words by finding the pair of senses with the greatest word overlap in their dictionary definitions (Lesk, 1986).

- Supervised methods. Supervised methods mainly adopt context to disambiguate words. A supervised method includes a training phase and a testing phase. In the training phase, a sense-annotated training corpus is required, from which syntactic and semantic features are extracted to create a classifier using machine

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 28–36,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

learning techniques, such as Support Vector Machine (Novischi et al., 2007). In the following testing phase, a word is classified into senses (Mihalcea, 2002) (Ng and Lee, 1996). Currently supervised methods achieve the best disambiguation quality (about 80% precision and recall for coarse-grained WSD in the most recent WSD evaluation conference SemEval 2007 (Navigli et al., 2007)). Nevertheless, since training corpora are manually annotated and expensive, supervised methods are often brittle due to data scarcity, and it is hard to annotate and acquire sufficient contextual information for every sense of a large number of words existing in natural languages.

- Semi-supervised methods. To overcome the knowledge acquisition bottleneck problem suffered by supervised methods, these methods make use of a small annotated corpus as seed data in a bootstrapping process (Hearst, 1991) (Yarowsky, 1995). A word-aligned bilingual corpus can also serve as seed data (Ng, Wang, and Chan, 2003).

- Unsupervised methods. These methods acquire contextual information directly from unannotated raw text, and senses can be induced from text using some similarity measure (Lin, 1997). However, automatically acquired information is often noisy or even erroneous. In the most recent SemEval 2007 (Navigli et al., 2007), the best unsupervised systems only achieved about 70% precision and 50% recall.

Disambiguation of a limited number of words is not hard, and necessary context information can be carefully collected and hand-crafted to achieve high disambiguation accuracy as shown in (Yarowsky, 1995). However, such approaches suffer a significant performance drop in practice when domain or vocabulary is not limited. Such a "cliff-style" performance collapse is called brittleness, which is due to insufficient knowledge and shared by many techniques in Artificial Intelligence. The main challenge of a WSD system is how to overcome the knowledge acquisition bottleneck and efficiently collect the huge amount of context knowledge. More precisely, a practical WSD need figure out how to create and maintain a comprehensive, dynamic, and up-to-date context knowledge base in a highly automatic manner. The context knowledge required in WSD has the following properties:

1. The context knowledge need cover a large number of words and their usage. Such a requirement of broad coverage is not trivial because a natural language usually contains thousands of words, and some popular words can have dozens of senses. For example, the Oxford English Dictionary has approximately 301,100 main entries (Oxford, 2003), and the average polysemy of the WordNet inventory is 6.18 (Fellbaum, 1998). Clearly acquisition of such a huge amount of knowledge can only be achieved with automatic techniques.

2. Natural language is not a static phenomenon. New usage of existing words emerges, which creates new senses. New words are created, and some words may "die" over time. It is estimated that every year around 2,500 new words appear in English (Kister, 1992). Such dynamics requires a timely maintenance and updating of context knowledge base, which makes manual collection even more impractical.

Taking into consideration the large amount and dynamic nature of context knowledge, we only have limited options when choosing knowledge sources for WSD. WSD is often an unconscious process to human beings. With a dictionary and sample sentences/phrases an average educated person can correctly disambiguate most polysemous words. Inspired by human WSD process, we choose an electronic dictionary and unannotated text samples of word instances as context knowledge sources for our WSD system. Both sources can be automatically accessed, provide an excellent coverage of word meanings and usage, and are actively updated to reflect the current state of languages. In this paper we present a fully unsupervised WSD system, which only requires WordNet sense inventory and unannotated text. In the rest of this paper, section 2 describes how to acquire and represent the context knowledge for WSD. We present our WSD algorithm in section 3. Our WSD system is evaluated with SemEval-2007 Task 7 (Coarse-grained English

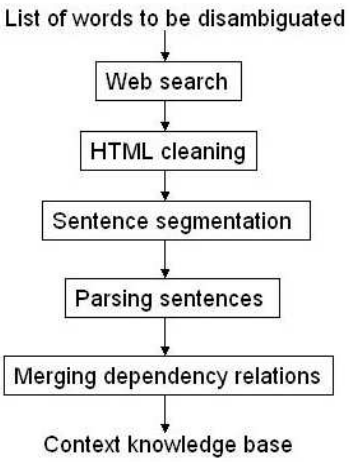


Figure 1: Context Knowledge Acquisition and Representation Process

All-words Task) data set, and the experiment results are discussed in section 4. We conclude in section 5.

2 Context Knowledge Acquisition and Representation

Figure 1 shows an overview of our context knowledge acquisition process, and collected knowledge is saved in a local knowledge base. Here are some details about each step.

2.1 Corpus building through Web search

The goal of this step is to collect as many as possible valid sample sentences containing the instances of to-be-disambiguated words. Preferably these instances are also diverse and cover many senses of a word. We have considered two possible text sources,

1. Electronic text collection, e.g., Gutenberg project (Gutenberg, 1971). Such collections often include thousands of books, which are often written by professionals and can provide many valid and accurate usage of a large number of words. Nevertheless, books in these collections are usually copyright-free and old, hence are lack of new words or new senses of words used in modern English.

2. Web documents. Billions of documents exist in the World Wide Web, and millions of Web pages are created and updated everyday. Such a huge dynamic text collection is an ideal source to provide broad and up-to-date context knowledge for WSD. The major concern about Web documents is inconsistency of their quality, and many Web pages are spam or contain erroneous information. However, factual errors in Web pages will not hurt the performance of WSD. Nevertheless, the quality of context knowledge is affected by broken sentences of poor linguistic quality and invalid word usage, e.g., sentences like "Colorless green ideas sleep furiously" that violate commonsense knowledge. Based on our experience these kind of errors are negligible when using popular Web search engines to retrieve relevant Web pages.

To start the acquisition process, words that need to be disambiguated are compiled and saved in a text file. Each single word is submitted to a Web search engine as a query. Several search engines provide API's for research communities to automatically retrieve large number of Web pages. In our experiments we used both Google and Yahoo! API's to retrieve up to 1,000 Web pages for each to-be-disambiguated word. Collected Web pages are cleaned first, e.g., control characters and HTML tags are removed. Then sentences are segmented simply based on punctuation (e.g., ?, !, .). Sentences that contain the instances of a specific word are extracted and saved into a local repository.

2.2 Parsing

Sentences organized according to each word are sent to a dependency parser, Minipar. Dependency parsers have been widely used in Computational Linguistics and natural language processing. An evaluation with the SUSANNE corpus shows that Minipar achieves 89% precision with respect to dependency relations (Lin, 1998). After parsing sentences are converted to parsing trees and saved in files. Neither our simple sentence segmentation approach nor Minipar parsing is 100% accurate, so a small number of invalid dependency relations may exist in parsing trees. The impact of these erroneous relations will be minimized in our WSD algorithm. Comparing with tagging or chunking, parsing is relatively expensive and time-consuming. However, in our method parsing is not performed in real time when we disambiguate words. Instead, sentences

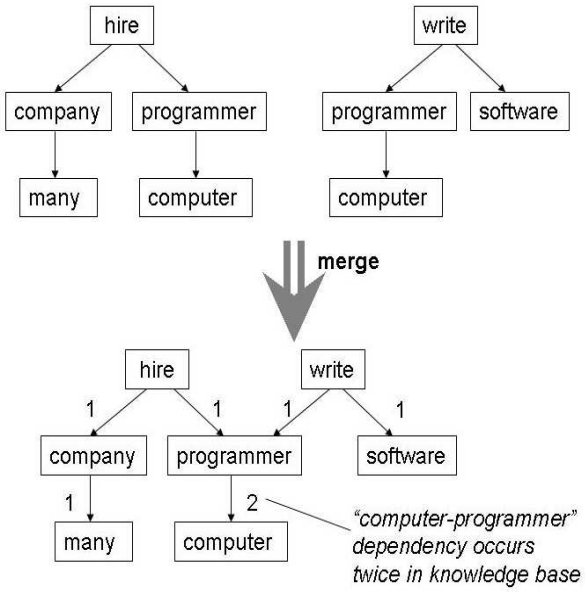


Figure 2: Merging two parsing trees. The number beside each edge is the number of occurrences of this dependency relation existing in the context knowledge base.

are parsed only once to extract dependency relations, then these relations are merged and saved in a local knowledge base for the following disambiguation. Hence, parsing will not affect the speed of disambiguation at all.

2.3 Merging dependency relations

After parsing, dependency relations from different sentences are merged and saved in a context knowledge base. The merging process is straightforward. A dependency relation includes one head word/node and one dependent word/node. Nodes from different dependency relations are merged into one as long as they represent the same word. An example is shown in Figure 2, which merges the following two sentences:

"Computer programmers write software."

"Many companies hire computer programmers."

In a dependency relation "$word_1 \rightarrow word_2$", $word_1$ is the head word, and $word_2$ is the dependent word. After merging dependency relations, we will obtain a weighted directed graph with a word as a node, a dependency relation as an edge, and the number of occurrences of dependency relation as weight of an edge. This weight indicates the strength of semantic relevancy of head word and dependent word. This graph will be used in the following WSD process as our context knowledge base. As a fully automatic knowledge acquisition process, it is inevitable to include erroneous dependency relations in the knowledge base. However, since in a large text collection valid dependency relations tend to repeat far more times than invalid ones, these erroneous edges only have minimal impact on the disambiguation quality as shown in our evaluation results.

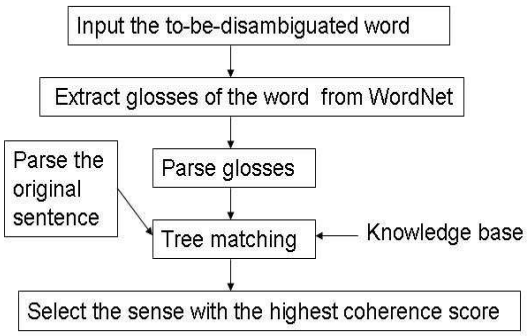


Figure 3: WSD Procedure

3 WSD Algorithm

Our WSD approach is based on the following insight:

If a word is semantically coherent with its context, then at least one sense of this word is semantically coherent with its context.

Assume that the text to be disambiguated is semantically valid, if we replace a word with its glosses one by one, the correct sense should be the one that will maximize the semantic coherence within this word's context. Based on this idea we set up our WSD procedure as shown in Figure 3. First both the original sentence that contains the to-be-disambiguated word and the glosses of to-be-disambiguated word are parsed. Then the parsing tree generated from each gloss is matched with the parsing tree of original sentence one by one. The gloss most semantically coherent with the original sentence will be chosen as the correct sense. How to measure the semantic coherence is critical. Our idea is based on the following hypotheses (assume $word_1$ is the to-be-disambiguated word):

- In a sentence if $word_1$ is dependent on $word_2$, and we denote the gloss of the correct sense of $word_1$ as g_{1i}, then g_{1i} contains the most semantically coherent words that are dependent

on $word_2$;

- In a sentence if a set of words DEP_1 are dependent on $word_1$, and we denote the gloss of the correct sense of $word_1$ as g_{1i}, then g_{1i} contains the most semantically coherent words that DEP_1 are dependent on.

For example, we try to disambiguate "company" in "A large company hires many computer programmers", after parsing we obtain the dependency relations "hire → company" and "company → large". The correct sense for the word "company" should be "an institution created to conduct business". If in the context knowledge base there exist the dependency relations "hire → institution" or "institution → large", then we believe that the gloss "an institution created to conduct business" is semantically coherent with its context - the original sentence. The gloss with the highest semantic coherence will be chosen as the correct sense. Obviously, the size of context knowledge base has a positive impact on the disambiguation quality, which is also verified in our experiments (see Section 4.2). Figure 4 shows our detailed WSD algorithm. Semantic coherence score is generated by the function $TreeMatching$, and we adopt a sentence as the context of a word.

We illustrate our WSD algorithm through an example. Assume we try to disambiguate "company" in the sentence "A large software company hires many computer programmers". "company" has 9 senses as a noun in WordNet 2.1. Let's pick the following two glosses to go through our WSD process.

- an institution created to conduct business
- small military unit

First we parse the original sentence and two glosses, and get three weighted parsing trees as shown in Figure 5. All weights are assigned to nodes/words in these parsing trees. In the parsing tree of the original sentence the weight of a node is reciprocal of the distance between this node and to-be-disambiguated node "company" (line 12 in Figure 4). In the parsing tree of a gloss the weight of a node is reciprocal of the level of this node in the parsing tree (line 16 in Figure 4). Assume that our context knowledge base contains relevant dependency relations shown in Figure 6.

Input: Glosses from WordNet;
S: the sentence to be disambiguated;
G: the knowledge base generated in Section 2;

1. Input a sentence S, W = $\{w|$ w's part of speech is noun, verb, adjective, or adverb, $w \in S\}$;
2. Parse S with a dependency parser, generate parsing tree T_S;
3. For each $w \in W$ {
4. Input all w's glosses from WordNet;
5. For each gloss w_i {
6. Parse w_i, get a parsing tree T_{wi};
7. score = TreeMatching(T_S, T_{wi});
   }
8. If the highest score is larger than a preset threshold, choose the sense with the highest score as the correct sense;
9. Otherwise, choose the first sense.
10. }

TreeMatching(T_S, T_{wi})
11. For each node $n_{Si} \in T_S$ {
12. Assign weight $w_{Si} = \frac{1}{l_{Si}}$, l_{Si} is the length between n_{Si} and w_i in T_S;
13. }
14. For each node $n_{wi} \in T_{wi}$ {
15. Load its dependent words D_{wi} from G;
16. Assign weight $w_{wi} = \frac{1}{l_{wi}}$, l_{wi} is the level number of n_{wi} in T_{wi};
17. For each n_{Sj} {
18. If $n_{Sj} \in D_{wi}$
19. calculate connection strength s_{ji} between n_{Sj} and n_{wi};
20. score = score + $w_{Si} \times w_{wi} \times s_{ji}$;
21. }
22. }
23. Return score;

Figure 4: WSD Algorithm

The weights in the context knowledge base are assigned to dependency relation edges. These weights are normalized to [0, 1] based on the number of dependency relation instances obtained in the acquisition and merging process. A large number of occurrences will be normalized to a high value (close to 1), and a small number of occurrences will be nor-

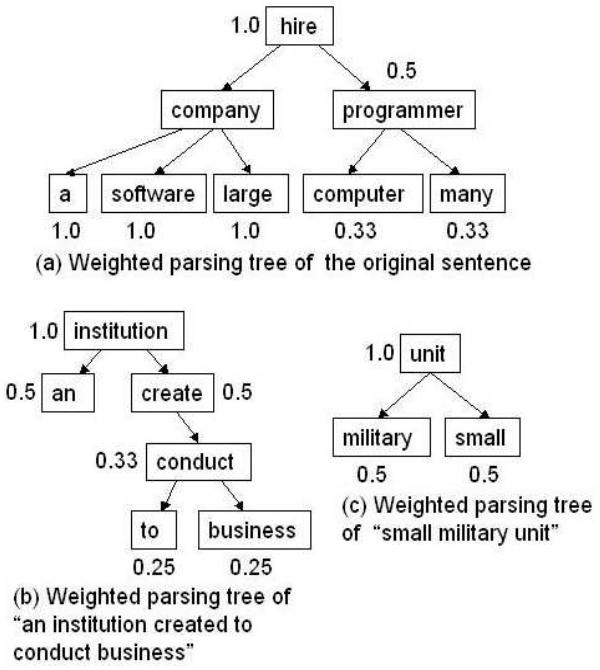


Figure 5: Weighted parsing trees of the original sentence and two glosses of "company"

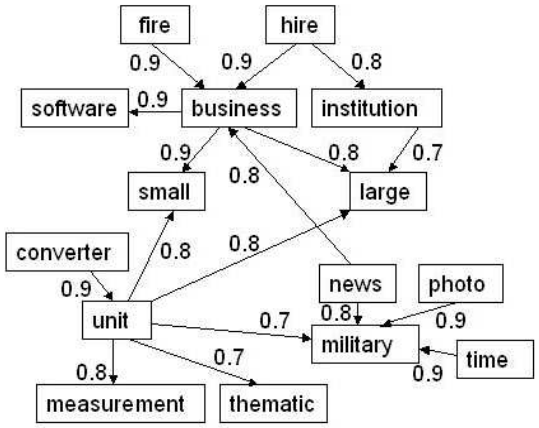


Figure 6: A fragment of context knowledge base

malized to a low value (close to 0).

Now we load the dependent words of each word in gloss 1 from the knowledge base (line 14, 15 in Figure 4), and we get {small, large} for "institution" and {large, software} for "business". In the dependent words of "company", "large" belongs to the dependent word sets of "institution" and "business", and "software" belongs to the dependent word set of "business", so the coherence score of gloss 1 is calculated as (line 19, 20 in Figure 4):

$1.0 \times 1.0 \times 0.7 + 1.0 \times 0.25 \times 0.8 + 1.0 \times 0.25 \times 0.9 = 1.125$

We go through the same process with the second gloss "small military unit". "Large" is the only dependent word of "company" appearing in the dependent word set of "unit" in gloss 2, so the coherence score of gloss 2 in the current context is:

$1.0 \times 1.0 \times 0.8 = 0.8$

After comparing the coherence scores of two glosses, we choose sense 1 of "company" as the correct sense (line 9 in Figure 4). This example illustrates that a strong dependency relation between a head word and a dependent word has a powerful disambiguation capability, and disambiguation quality is also significantly affected by the quality of dictionary definitions.

In Figure 4 the $TreeMatching$ function matches the dependent words of to-be-disambiguated word (line 15 in Figure 4), and we call this matching strategy as dependency matching. This strategy will not work if a to-be-disambiguated word has no dependent words at all, for example, when the word "company" in "Companies hire computer programmers" has no dependent words. In this case, we developed the second matching strategy, which is to match the head words that the to-be-disambiguated word is dependent on, such as matching "hire" (the head word of "company") in Figure 5(a). Using the dependency relation "hire → company", we can correctly choose sense 1 since there is no such relation as "hire → unit" in the knowledge base. This strategy is also helpful when disambiguating adjectives and adverbs since they usually only depend on other words, and rarely any other words are dependent on them. The third matching strategy is to consider synonyms as a match besides the exact matching words. Synonyms can be obtained through the synsets in WordNet. For example, when we disambiguate "company" in "Big companies hire many computer programmers", "big" can be considered as a match for "large". We call this matching strategy as synonym matching. The three matching strategies can be combined and applied together, and in Section 4.1 we show the experiment results of 5 different matching strategy combinations.

4 Experiments

We have evaluated our method using SemEval-2007 Task 07 (Coarse-grained English All-words Task) test set (Navigli et al., 2007). The task organizers provide a coarse-grained sense inventory created with SSI algorithm (Navigli and Velardi, 2005), training data, and test data. Since our method does not need any training or special tuning, neither coarse-grained sense inventory nor training data was used. The test data includes: a news article about "homeless" (including totally 951 words, 368 words are annotated and need to be disambiguated), a review of the book "Feeding Frenzy" (including totally 987 words, 379 words are annotated and need to be disambiguated), an article about some traveling experience in France (including totally 1311 words, 500 words are annotated and need to be disambiguated), computer programming(including totally 1326 words, 677 words are annotated and need to be disambiguated), and a biography of the painter Masaccio (including totally 802 words, 345 words are annotated and need to be disambiguated). Two authors of (Navigli et al., 2007) independently and manually annotated part of the test set (710 word instances), and the pairwise agreement was 93.80%. This inter-annotator agreement is usually considered an upper-bound for WSD systems.

We followed the WSD process described in Section 2 and 3 using the WordNet 2.1 sense repository that is adopted by SemEval-2007 Task 07. All experiments were performed on a Pentium 2.33GHz dual core PC with 3GB memory. Among the 2269 to-be-disambiguated words in the five test documents, 1112 words are unique and submitted to Google API as queries. The retrieved Web pages were cleaned, and 1945189 relevant sentences were extracted. On average 1749 sentences were obtained for each word. The Web page retrieval step took 3 days, and the cleaning step took 2 days. Parsing was very time-consuming and took 11 days. The merging step took 3 days. Disambiguation of 2269 words in the 5 test articles took 4 hours. All these steps can be parallelized and run on multiple computers, and the whole process will be shortened accordingly.

The overall disambiguation results are shown in Table 1. For comparison we also listed the results of the top three systems and three unsupervised systems participating in SemEval-2007 Task 07. All of the top three systems (UoR-SSI, NUS-PT, NUS-ML) are supervised systems, which used annotated resources (e.g., SemCor, Defense Science Organization Corpus) during the training phase. Our fully unsupervised WSD system significantly outperforms the three unsupervised systems (SUSSZ-FR, SUSSX-C-WD, SUSSX-CR) and achieves performance approaching the top-performing supervised WSD systems.

4.1 Impact of different matching strategies to disambiguation quality

To test the effectiveness of different matching strategies discussed in Section 3, we performed some additional experiments. Table 2 shows the disambiguation results by each individual document with the following 5 matching strategies:

1. Dependency matching only.
2. Dependency and backward matching.
3. Dependency and synonym backward matching.
4. Dependency and synonym dependency matching.
5. Dependency, backward, synonym backward, and synonym dependency matching.

As expected combination of more matching strategies results in higher disambiguation quality. By analyzing the scoring details, we verified that backward matching is especially useful to disambiguate adjectives and adverbs. Adjectives and adverbs are often dependent words, so dependency matching itself rarely finds any matched words. Since synonyms are semantically equivalent, it is reasonable that synonym matching can also improve disambiguation performance.

4.2 Impact of knowledge base size to disambiguation quality

To test the impact of knowledge base size to disambiguation quality we randomly selected 1339264 sentences (about two thirds of all sentences) from our text collection and built a smaller knowledge base. Table 3 shows the experiment results. Overall disambiguation quality has dropped slightly, which

System	Attempted	Precision	Recall	F1
UoR-SSI	100.0	83.21	83.21	83.21
NUS-PT	100.0	82.50	82.50	82.50
NUS-ML	100.0	81.58	81.58	81.58
TreeMatch	**100.0**	**73.65**	**73.65**	**73.65**
SUSSZ-FR	72.8	71.73	52.23	60.44
SUSSX-C-WD	72.8	54.54	39.71	45.96
SUSSX-CR	72.8	54.30	39.53	45.75

Table 1: Overall disambiguation scores (Our system "TreeMatch" is marked in bold)

Matching strategy	d001		d002		d003		d004		d005		Overall	
	P	R	P	R	P	R	P	R	P	R	P	R
1	72.28	72.28	66.23	66.23	63.20	63.20	66.47	66.47	56.52	56.52	65.14	65.14
2	70.65	70.65	70.98	70.98	65.20	65.20	72.23	72.23	58.84	58.84	68.18	68.18
3	79.89	79.89	75.20	75.20	69.00	69.00	71.94	71.94	64.64	64.64	72.01	72.01
4	80.71	80.71	78.10	78.10	72.80	72.80	71.05	71.05	67.54	67.54	73.65	73.65
5	80.16	80.16	78.10	78.10	69.40	69.40	72.82	72.82	66.09	66.09	73.12	73.12

Table 2: Disambiguation scores by article with 5 matching strategies

shows a positive correlation between the amount of context knowledge and disambiguation quality. It is reasonable to assume that our disambiguation performance can be improved further by collecting and incorporating more context knowledge.

Matching strategy	Overall	
	P	R
1	65.36	65.36
2	67.78	67.78
3	68.09	68.09
4	70.69	70.69
5	67.78	67.78

Table 3: Disambiguation scores by article with a smaller knowledge base

5 Conclusion and Future Work

Broad coverage and disambiguation quality are critical for WSD techniques to be adopted in practice. This paper proposed a fully unsupervised WSD method. We have evaluated our approach with SemEval-2007 Task 7 (Coarse-grained English All-words Task) data set, and we achieved F-scores approaching the top performing supervised WSD systems. By using widely available unannotated text and a fully unsupervised disambiguation approach, our method may provide a viable solution to the problem of WSD. The future work includes:

1. Continue to build the knowledge base, enlarge the coverage and improve the system performance. The experiment results in Section 4.2 clearly show that more word instances can improve the disambiguation accuracy and recall scores;

2. WSD is often an unconscious process for human beings. It is unlikely that a reader examines all surrounding words when determining the sense of a word, which calls for a smarter and more selective matching strategy than what we have tried in Section 4.1;

3. Test our WSD system on fine-grained SemEval 2007 WSD task 17. Although we only evaluated our approach with coarse-grained senses, our method can be directly applied to fine-grained WSD without any modifications.

Acknowledgments

This work is partially funded by NSF grant 0737408 and Scholar Academy at the University of Houston Downtown. This paper contains proprietary information protected under a pending U.S. patent.

References

Agirre, Eneko, Philip Edmonds (eds.). 2006. Word Sense Disambiguation: Algorithms and Applications, Springer.

Chklovski, T. and Mihalcea, R. 2002. Building a sense tagged corpus with open mind word expert. In Proceedings of the Acl-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions, Morristown, NJ, 116-122.

C. Fellbaum, WordNet: An Electronic Lexical Database, MIT press, 1998

Project Gutenberg, available at www.gutenberg.org

Hearst, M. (1991) Noun Homograph Disambiguation Using Local Context in Large Text Corpora, Proc. 7th Annual Conference of the University of Waterloo Center for the New OED and Text Research, Oxford.

Nancy Ide and Jean Véronis. 1998. Introduction to the special issue on word sense disambiguation: the state of the art. *Comput. Linguist.*, 24(1):2–40.

Kister, Ken. "Dictionaries defined", Library Journal, Vol. 117 Issue 11, p43, 4p, 2bw

Lesk, M. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In Proceedings of the 5th Annual international Conference on Systems Documentation (Toronto, Ontario, Canada). V. DeBuys, Ed. SIGDOC '86.

Dekang Lin. 1998. Dependency-based evaluation of minipar. In *Proceedings of the LREC Workshop on the Evaluation of Parsing Systems*, pages 234–241, Granada, Spain.

Lin, D. 1997. Using syntactic dependency as local context to resolve word sense ambiguity. In Proceedings of the 35th Annual Meeting of the Association For Computational Linguistics and Eighth Conference of the European Chapter of the Association For Computational Linguistics (Madrid, Spain, July 07 - 12, 1997).

Rada Mihalcea, Using Wikipedia for Automatic Word Sense Disambiguation, in Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2007), Rochester, April 2007.

Rada Mihalcea. 2002. Instance based learning with automatic feature selection applied to word sense disambiguation. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ.

Dan Moldovan and Vasile Rus, Explaining Answers with Extended WordNet, ACL 2001.

Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. 2007. Semeval-2007 task 07: Coarse-grained english all-words task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 30–35, Prague, Czech Republic.

Roberto Navigli and Paola Velardi. 2005. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 27(7):10631074.

Hwee Tou Ng, Bin Wang, and Yee Seng Chan. Exploiting Parallel Texts for Word Sense Disambiguation: An Empirical Study. ACL, 2003.

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: an exemplar-based approach. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 40–47, Morristown, NJ.

Adrian Novischi, Muirathnam Srikanth, and Andrew Bennett. 2007. Lcc-wsd: System description for English coarse grained all words task at semeval 2007. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 223–226, Prague, Czech Republic.

Catherine Soanes and Angus Stevenson, editors. 2003. Oxford Dictionary of English. Oxford University Press.

Rada Mihalcea, available at http://www.cs.unt.edu/ rada/downloads.html

Yarowsky, D. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In Proceedings of the 33rd Annual Meeting on Association For Computational Linguistics (Cambridge, Massachusetts, June 26 - 30, 1995).

Learning Phoneme Mappings for Transliteration without Parallel Data

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

Abstract

We present a method for performing machine transliteration without any parallel resources. We frame the transliteration task as a decipherment problem and show that it is possible to learn cross-language phoneme mapping tables using only monolingual resources. We compare various methods and evaluate their accuracies on a standard name transliteration task.

1 Introduction

Transliteration refers to the transport of names and terms between languages with different writing systems and phoneme inventories. Recently there has been a large amount of interesting work in this area, and the literature has outgrown being citable in its entirety. Much of this work focuses on *back-transliteration*, which tries to restore a name or term that has been transported into a foreign language. Here, there is often only one correct target spelling—for example, given `jyon.kairu` (the name of a U.S. Senator transported to Japanese), we must output "Jon Kyl", not "John Kyre" or any other variation.

There are many techniques for transliteration and back-transliteration, and they vary along a number of dimensions:

- phoneme substitution vs. character substitution
- heuristic vs. generative vs. discriminative models
- manual vs. automatic knowledge acquisition

We explore the third dimension, where we see several techniques in use:

- Manually-constructed transliteration models, e.g., (Hermjakob et al., 2008).
- Models constructed from bilingual dictionaries of terms and names, e.g., (Knight and Graehl, 1998; Huang et al., 2004; Haizhou et al., 2004; Zelenko and Aone, 2006; Yoon et al., 2007; Li et al., 2007; Karimi et al., 2007; Sherif and Kondrak, 2007b; Goldwasser and Roth, 2008b).
- Extraction of parallel examples from bilingual corpora, using bootstrap dictionaries e.g., (Sherif and Kondrak, 2007a; Goldwasser and Roth, 2008a).
- Extraction of parallel examples from comparable corpora, using bootstrap dictionaries, and temporal and word co-occurrence, e.g., (Sproat et al., 2006; Klementiev and Roth, 2008).
- Extraction of parallel examples from web queries, using bootstrap dictionaries, e.g., (Nagata et al., 2001; Oh and Isahara, 2006; Kuo et al., 2006; Wu and Chang, 2007).
- Comparing terms from different languages in phonetic space, e.g., (Tao et al., 2006; Goldberg and Elhadad, 2008).

In this paper, we investigate methods to acquire transliteration mappings *from non-parallel sources*. We are inspired by previous work in unsupervised learning for natural language, e.g. (Yarowsky, 1995;

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 37–45,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

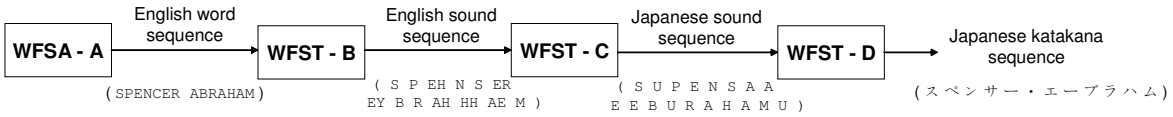


Figure 1: Model used for back-transliteration of Japanese katakana names and terms into English. The model employs a four-stage cascade of weighted finite-state transducers (Knight and Graehl, 1998).

Goldwater and Griffiths, 2007), and we are also inspired by cryptanalysis—we view a corpus of foreign terms as a code for English, and we attempt to break the code.

2 Background

We follow (Knight and Graehl, 1998) in tackling back-transliteration of Japanese katakana expressions into English. Knight and Graehl (1998) developed a four-stage cascade of finite-state transducers, shown in Figure 1.

- **WFSA A** - produces an English word sequence w with probability P(w) (based on a unigram word model).
- **WFST B** - generates an English phoneme sequence e corresponding to w with probability P($e|w$).
- **WFST C** - transforms the English phoneme sequence into a Japanese phoneme sequence j according to a model P($j|e$).
- **WFST D** - writes out the Japanese phoneme sequence into Japanese katakana characters according to a model P($k|j$).

Using the cascade in the reverse (noisy-channel) direction, they are able to translate new katakana names and terms into English. They report 36% error in translating 100 U.S. Senators' names, and they report exceeding human transliteration performance in the presence of optical scanning noise.

The only transducer that requires parallel training data is WFST C. Knight and Graehl (1998) take several thousand phoneme string pairs, automatically align them with the EM algorithm (Dempster et al., 1977), and construct WFST C from the aligned phoneme pieces.

We re-implement their basic method by instantiating a densely-connected version of WFST C with all 1-to-1 and 1-to-2 phoneme connections between English and Japanese. Phoneme bigrams that occur fewer than 10 times in a Japanese corpus are omitted, and we omit 1-to-3 connections. This initial WFST C model has 15320 uniformly weighted parameters. We then train the model on 3343 phoneme string pairs from a bilingual dictionary, using the EM algorithm. EM immediately reduces the connections in the model to those actually observed in the parallel data, and after 14 iterations, there are only 188 connections left with P($j|e$) ≥ 0.01. Figure 2 shows the phonemic substitution table learnt from parallel training.

We use this trained WFST C model and apply it to the U.S. Senator name transliteration task (which we update to the 2008 roster). We obtain 40% error, roughly matching the performance observed in (Knight and Graehl, 1998).

3 Task and Data

The task of this paper is to learn the mappings in Figure 2, *but without parallel data*, and to test those mappings in end-to-end transliteration. We imagine our problem as one faced by monolingual English speaker wandering around Japan, reading a multitude of katakana signs, listening to people speak Japanese, and eventually deciphering those signs into English. To mis-quote Warren Weaver:

> "When I look at a corpus of Japanese katakana, I say to myself, this is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."

Our larger motivation is to move toward easily-built transliteration systems for all language pairs, regardless of parallel resources. While Japanese/English transliteration has its own particular features, we believe it is a reasonable starting point.

e	j	P(j|e)	e	j	P(j|e)	e	j	P(j|e)	e	j	P(j|e)	e	j	P(j|e)	e	j	P(j|e)	e	j	P(j|e)
AA	**o**	**0.49**	**AY**	**a i**	**0.84**	**EH**	**e**	**0.94**	**HH**	**h**	**0.95**	**L**	**r**	**0.62**	**OY**	**o i**	**0.89**	**SH**	**sh y**	**0.33**	**V**	**b**	**0.75**
	a	**0.46**		**i**	**0.09**		**a**	**0.03**		**w**	**0.02**		**r u**	**0.37**		**o e**	**0.04**		**sh**	**0.31**		**b u**	**0.17**
	o o	**0.02**		**a**	**0.03**					**h a**	**0.02**					**o**	**0.04**		**y u**	**0.17**		**w**	**0.03**
	a a	**0.02**		**i y**	**0.01**											**i**	**0.04**		**ssh y**	**0.12**		**a**	**0.02**
				a y	**0.01**														**sh i**	**0.04**			
																			ssh	**0.02**			
																			e	**0.01**			
AE	**a**	**0.93**	**B**	**b**	**0.82**	**ER**	**a a**	**0.8**	**IH**	**i**	**0.89**	**M**	**m**	**0.68**	**P**	**p**	**0.63**	**T**	**t**	**0.43**	**W**	**w**	**0.73**
	a ssh	**0.02**		**b u**	**0.15**		**a**	**0.08**		**e**	**0.05**		**m u**	**0.22**		**p u**	**0.16**		**t o**	**0.25**		**u**	**0.17**
	a n	**0.02**					**a r**	**0.03**		**i n**	**0.01**		**n**	**0.08**		**pp u**	**0.13**		**tt o**	**0.17**		**o**	**0.04**
							r u	**0.02**		**a**	**0.01**					**pp**	**0.06**		**ts**	**0.04**		**i**	**0.02**
							o r	**0.02**											**tt**	**0.03**			
							e r	**0.02**											**u**	**0.02**			
																			ts u	**0.02**			
																			ch	**0.02**			
AH	**a**	**0.6**	**CH**	**tch i**	**0.27**	**EY**	**e e**	**0.58**	**IY**	**i i**	**0.58**	**N**	**n**	**0.96**	**PAUSE**	**pause**	**1.0**	**TH**	**s u**	**0.48**	**Y**	**y**	**0.7**
	o	**0.13**		**ch**	**0.24**		**e**	**0.15**		**i**	**0.3**		**nn**	**0.02**					**s**	**0.22**		**i**	**0.26**
	e	**0.11**		**ch i**	**0.23**		**e i**	**0.12**		**e**	**0.07**								**sh**	**0.16**		**e**	**0.02**
	i	**0.07**		**ch y**	**0.2**		**a**	**0.1**		**e e**	**0.03**								**t o**	**0.04**		**a**	**0.02**
	u	**0.06**		**tch y**	**0.02**		**a i**	**0.03**											**ch**	**0.04**			
				tch	**0.02**														**t e**	**0.02**			
				ssh y	**0.01**														**t**	**0.02**			
				k	**0.01**														**a**	**0.02**			
AO	**o**	**0.6**	**D**	**d**	**0.54**	**F**	**h**	**0.58**	**JH**	**j y**	**0.35**	**NG**	**n**	**0.62**	**R**	**r**	**0.61**	**UH**	**u**	**0.79**	**Z**	**z**	**0.27**
	o o	**0.27**		**d o**	**0.27**		**h u**	**0.35**		**j**	**0.24**		**g u**	**0.22**		**a**	**0.27**		**u u**	**0.09**		**z u**	**0.25**
	a	**0.05**		**dd o**	**0.06**		**hh**	**0.04**		**j i**	**0.21**		**n g**	**0.09**		**o**	**0.07**		**u a**	**0.04**		**u**	**0.16**
	o n	**0.03**		**z**	**0.02**		**hh u**	**0.02**		**jj i**	**0.14**		**i**	**0.04**		**r u**	**0.03**		**dd**	**0.03**		**s u**	**0.07**
	a u	**0.03**		**j**	**0.02**					**z**	**0.04**		**u**	**0.01**		**a a**	**0.01**		**u ssh**	**0.02**		**j**	**0.06**
	u	**0.01**		**u**	**0.01**								**o**	**0.01**					**o**	**0.02**		**a**	**0.06**
													a	**0.01**								**n**	**0.03**
																						i	**0.03**
																						s	**0.02**
																						o	**0.02**
AW	**a u**	**0.69**	**DH**	**z**	**0.87**	**G**	**g**	**0.66**	**K**	**k**	**0.53**	**OW**	**o**	**0.57**	**S**	**s u**	**0.43**	**UW**	**u u**	**0.67**	**ZH**	**j y**	**0.43**
	a w	**0.15**		**z u**	**0.08**		**g u**	**0.19**		**k u**	**0.2**		**o o**	**0.39**		**s**	**0.37**		**u**	**0.29**		**j i**	**0.29**
	a o	**0.06**		**a z**	**0.04**		**gg u**	**0.1**		**kk u**	**0.16**		**o u**	**0.02**		**sh**	**0.08**		**y u**	**0.02**		**j**	**0.29**
	a	**0.04**					**g y**	**0.03**		**kk**	**0.05**					**u**	**0.05**						
	u u	**0.02**					**gg**	**0.01**		**k y**	**0.02**					**ss**	**0.02**						
	o o	**0.02**					**g a**	**0.01**		**k i**	**0.01**					**ssh**	**0.01**						
	o	**0.02**																					

Figure 2: Phonemic substitution table learnt from 3343 parallel English/Japanese phoneme string pairs. English phonemes are in uppercase, Japanese in lowercase. Mappings with P(j\|e) > 0.01 are shown.

A A CH I D O	CH E N J I	N E B A D A	W A K O B I A
A A K U P U R A Z A	CH E S	:	W A N K A PP U
A A N D O	:	O P U T I K U S U	W A N T E N P O
A A T I S U T O	D E K O R A T I B U	:	W A S E R I N
A A T O S E R I N A P I S U T O N	D E T O M O R U T O	P I I T A A	Y U N I O N
A I A N B I R U	E P I G U R A M U	P I KK A A	Y U N I TT O SH I S U T E M U
A I D I I D O	E R A N D O	P I N G U U	Y U U
A I K E N B E R I I	:	P I P E R A J I N A M I D O	:
A J I A K A PP U	J Y A I A N TS U	P I S A	:
A J I T O	J Y A Z U	P I U R A	Z E N E R A R U E A K O N
A K A SH I A K O O S U	:	P O I N T O	Z E R O
A K U A	M Y U U Z E U M U	:	Z O N B I I Z U
:	:	:	:
:	:	:	:

Figure 3: Some Japanese phoneme sequences generated from the monolingual katakana corpus using WFST D.

Our monolingual resources are:

- 43717 unique Japanese katakana sequences collected from web newspaper data. We split multi-word katakana phrases on the center-dot ("·") character, and select a final corpus of 9350 unique sequences. We add monolingual Japanese versions of the 2008 U.S. Senate roster.[1]
- The CMU pronunciation dictionary of English, with 112,151 entries.
- The English gigaword corpus. Knight and Graehl (1998) already use frequently-occurring capitalized words to build the WFSA A component of their four-stage cascade.

We seek to use our English knowledge (derived from 2 and 3) to decipher the Japanese katakana corpus (1) into English. Figure 3 shows a portion of the Japanese corpus, which we transform into Japanese phoneme sequences using the monolingual resource of WFST D. We note that the Japanese phoneme inventory contains 39 unique ("ciphertext") symbols,

[1] We use "open" EM testing, in which unlabeled test data is allowed to be part of unsupervised training. However, no parallel data is allowed.

compared to the 40 English ("plaintext") phonemes.

Our goal is to compare and evaluate the WFST C model learnt under two different scenarios—(a) using parallel data, and (b) using monolingual data. For each experiment, we train only the WFST C model and then apply it to the name transliteration task—decoding 100 U.S. Senator names from Japanese to English using the automata shown in Figure 1. For all experiments, we keep the rest of the models in the cascade (WFSA A, WFST B, and WFST D) unchanged. We evaluate on whole-name error-rate (maximum of 100/100) as well as normalized word edit distance, which gives partial credit for getting the first or last name correct.

4 Acquiring Phoneme Mappings from Non-Parallel Data

Our main data consists of 9350 unique Japanese phoneme sequences, which we can consider as a single long sequence j. As suggested by Knight et al (2006), we explain the existence of j as the result of someone initially producing a long English phoneme sequence e, according to P(e), then transforming it into j, according to P($j|e$). The probability of our observed data P(j) can be written as:

$$P(j) = \sum_{e} P(e) \cdot P(j|e)$$

We take P(e) to be some fixed model of monolingual English phoneme production, represented as a weighted finite-state acceptor (WFSA). P($j|e$) is implemented as the initial, uniformly-weighted WFST C described in Section 2, with 15320 phonemic connections.

We next maximize P(j) by manipulating the substitution table P($j|e$), aiming to produce a result such as shown in Figure 2. We accomplish this by composing the English phoneme model P(e) WFSA with the P($j|e$) transducer. We then use the EM algorithm to train just the P($j|e$) parameters (inside the composition that predicts j), and guess the values for the individual phonemic substitutions that maximize the likelihood of the observed data P(j).[2]

[2]In our experiments, we use the Carmel finite-state transducer package (Graehl, 1997), a toolkit with an algorithm for EM training of weighted finite-state transducers.

We allow EM to run until the P(j) likelihood ratio between subsequent training iterations reaches 0.9999, and we terminate early if 200 iterations are reached.

Finally, we decode our test set of U.S. Senator names. Following Knight et al (2006), we stretch out the P($j|e$) model probabilities after decipherment training and prior to decoding our test set, by cubing their values.

Decipherment under the conditions of transliteration is substantially more difficult than solving letter-substitution ciphers (Knight et al., 2006; Ravi and Knight, 2008; Ravi and Knight, 2009) or phoneme-substitution ciphers (Knight and Yamada, 1999). This is because the target table contains significant non-determinism, and because each symbol has multiple possible fertilities, which introduces uncertainty about the length of the target string.

4.1 Baseline P(e) Model

Clearly, we can design P(e) in a number of ways. We might expect that the more the system knows about English, the better it will be able to decipher the Japanese. Our baseline P(e) is a 2-gram phoneme model trained on phoneme sequences from the CMU dictionary. The second row (2a) in Figure 4 shows results when we decipher with this fixed P(e). This approach performs poorly and gets all the Senator names wrong.

4.2 Consonant Parity

When training under non-parallel conditions, we find that we would like to keep our WFST C model small, rather than instantiating a fully-connected model. In the supervised case, parallel training allows the trained model to retain only those connections which were observed from the data, and this helps eliminate many bad connections from the model. In the unsupervised case, there is no parallel data available to help us make the right choices.

We therefore use prior knowledge and place a *consonant-parity* constraint on the WFST C model. Prior to EM training, we throw out any mapping from the P($j|e$) substitution model that does not have the same number of English and Japanese consonant phonemes. This is a pattern that we observe across a range of transliteration tasks. Here are ex-

	Phonemic Substitution Model	Name Transliteration Error	
		whole-name error	norm. edit distance
1	$e \rightarrow j$ = { 1-to-1, 1-to-2 } + EM aligned with parallel data	40	25.9
2a	$e \rightarrow j$ = { 1-to-1, 1-to-2 } + decipherment training with 2-gram English P(e)	100	100.0
2b	$e \rightarrow j$ = { 1-to-1, 1-to-2 } + decipherment training with 2-gram English P(e) + consonant-parity	98	89.8
2c	$e \rightarrow j$ = { 1-to-1, 1-to-2 } + decipherment training with 3-gram English P(e) + consonant-parity	94	73.6
2d	$e \rightarrow j$ = { 1-to-1, 1-to-2 } + decipherment training with a word-based English model + consonant-parity	77	57.2
2e	$e \rightarrow j$ = { 1-to-1, 1-to-2 } + decipherment training with a word-based English model + consonant-parity + initialize mappings having consonant matches with higher probability weights	73	54.2

Figure 4: Results on name transliteration obtained when using the phonemic substitution model trained under different scenarios—(1) parallel training data, (2a-e) using only monolingual resources.

amples of mappings where consonant parity is violated:

```
K  => a        N  => e e
EH => s a      EY => n
```

Modifying the WFST C in this way leads to better decipherment tables and slightly better results for the U.S. Senator task. Normalized edit distance drops from 100 to just under 90 (row 2b in Figure 4).

4.3 Better English Models

Row 2c in Figure 4 shows decipherment results when we move to a 3-gram English phoneme model for P(e). We notice considerable improvements in accuracy. On the U.S. Senator task, normalized edit distance drops from 89.8 to 73.6, and whole-name error decreases from 98 to 94.

When we analyze the results from deciphering with a 3-gram P(e) model, we find that many of the Japanese phoneme test sequences are decoded into English phoneme sequences (such as "`IH K R IH N`" and "`AE G M AH N`") that are not valid words. This happens because the models we used for decipherment so far have no knowledge of what constitutes a globally valid English sequence. To help the phonemic substitution model learn this information automatically, we build a word-based P(e) from English phoneme sequences in the CMU dictionary and use this model for decipherment training. The word-based model produces complete English phoneme sequences corresponding to 76,152 actual English words from the CMU dictionary. The English phoneme sequences are represented as paths through a WFSA, and all paths are weighted equally. We represent the word-based model in compact form, using determinization and minimization techniques applicable to weighted finite-state automata. This allows us to perform efficient EM training on the cascade of P(e) and P($j|e$) models. Under this scheme, English phoneme sequences resulting from decipherment are always analyzable into actual words.

Row 2d in Figure 4 shows the results we obtain when training our WFST C with a word-based English phoneme model. Using the word-based model produces the best result so far on the phonemic substitution task with non-parallel data. On the U.S. Senator task, word-based decipherment outperforms the other methods by a large margin. It gets 23 out of 100 Senator names exactly right, with a much lower normalized edit distance (57.2). We have managed to achieve this performance using only monolingual data. This also puts us within reach of the parallel-trained system's performance (40% whole-name errors, and 25.9 word edit distance error) without using a single English/Japanese pair for training.

To summarize, the quality of the English phoneme

e	j	P(j|e)	e	j	P(j|e)	e	j	P(j|e)	e	j	P(j|e)	e	j	P(j|e)	e	j	P(j|e)	e	j	P(j|e)
AA	**a**	**0.37**	**AY**	**a i**	**0.36**	**EH**	**e**	**0.37**	**HH**	**h**	**0.45**	**L**	**r**	**0.3**	**OY**	**a**	**0.27**	**SH**	**sh y**	**0.22**	**V**	**b**	**0.34**
	o	**0.25**		**o o**	**0.13**		**a**	**0.24**		**s**	**0.12**		**n**	**0.19**		**i**	**0.16**		**m**	**0.11**		**k**	**0.14**
	i	**0.15**		**e**	**0.12**		**o**	**0.12**		**k**	**0.09**		**r u**	**0.15**		**y u**	**0.1**		**r**	**0.1**		**m**	**0.13**
	u	**0.08**		**i**	**0.11**		**i**	**0.12**		**b**	**0.08**		**r i**	**0.04**		**o i**	**0.1**		**s**	**0.06**		**s**	**0.07**
	e	**0.07**		**a**	**0.11**		**u**	**0.06**		**m**	**0.07**		**t**	**0.03**		**y a**	**0.09**		**p**	**0.06**		**d**	**0.07**
	o o	**0.03**		**u u**	**0.05**		**o o**	**0.04**		**w**	**0.03**		**m u**	**0.02**		**y o**	**0.08**		**s a**	**0.05**		**r**	**0.04**
	y a	**0.01**		**y u**	**0.02**		**y u**	**0.01**		**p**	**0.03**		**m**	**0.02**		**e**	**0.08**		**h**	**0.05**		**t**	**0.03**
	a a	**0.01**		**u**	**0.02**		**a i**	**0.01**		**g**	**0.03**		**w a**	**0.01**		**o**	**0.06**		**b**	**0.05**		**h**	**0.02**
				o	**0.02**					**k y**	**0.02**		**t a**	**0.01**		**o o**	**0.02**		**t**	**0.04**		**sh**	**0.01**
				e e	**0.02**					**d**	**0.02**		**r a**	**0.01**		**e i**	**0.02**		**k**	**0.04**		**n**	**0.01**
AE	**a**	**0.52**	**B**	**b**	**0.41**	**ER**	**a a**	**0.47**	**IH**	**i**	**0.36**	**M**	**m**	**0.3**	**P**	**p**	**0.18**	**T**	**t**	**0.2**	**W**	**w**	**0.23**
	i	**0.19**		**p**	**0.12**		**a**	**0.17**		**e**	**0.25**		**n**	**0.08**		**p u**	**0.08**		**t o**	**0.16**		**r**	**0.2**
	e	**0.11**		**k**	**0.09**		**u**	**0.08**		**a**	**0.15**		**k**	**0.08**		**n**	**0.05**		**t a**	**0.05**		**m**	**0.13**
	o	**0.08**		**m**	**0.07**		**o**	**0.07**		**u**	**0.09**		**r**	**0.07**		**k**	**0.05**		**n**	**0.04**		**s**	**0.08**
	u	**0.03**		**s**	**0.04**		**e**	**0.04**		**o**	**0.09**		**s**	**0.06**		**sh i**	**0.04**		**k u**	**0.03**		**k**	**0.07**
	u u	**0.02**		**g**	**0.04**		**o o**	**0.03**		**o o**	**0.01**		**h**	**0.05**		**k u**	**0.04**		**k**	**0.03**		**h**	**0.06**
	o o	**0.02**		**t**	**0.03**		**i i**	**0.03**					**t**	**0.04**		**s u**	**0.03**		**t e**	**0.02**		**b**	**0.06**
				z	**0.02**		**y u**	**0.02**					**g**	**0.04**		**p a**	**0.03**		**s**	**0.02**		**t**	**0.04**
				d	**0.02**		**u u**	**0.02**					**b**	**0.04**		**t**	**0.02**		**r**	**0.02**		**p**	**0.04**
				ch y	**0.02**		**i**	**0.02**					**m u**	**0.03**		**m a**	**0.02**		**g u**	**0.02**		**d**	**0.02**
AH	**a**	**0.31**	**CH**	**g**	**0.12**	**EY**	**e e**	**0.3**	**IY**	**i**	**0.25**	**N**	**n**	**0.56**	**PAUSE**	**pause**	**1.0**	**TH**	**k**	**0.21**	**Y**	**s**	**0.25**
	o	**0.23**		**k**	**0.11**		**a**	**0.22**		**i i**	**0.21**		**r u**	**0.09**					**p u**	**0.11**		**k**	**0.18**
	i	**0.17**		**b**	**0.09**		**i**	**0.11**		**a**	**0.15**		**s u**	**0.04**					**k u**	**0.1**		**m**	**0.07**
	e	**0.12**		**sh**	**0.07**		**u**	**0.09**		**a a**	**0.12**		**m u**	**0.02**					**d**	**0.08**		**g**	**0.06**
	u	**0.1**		**s**	**0.07**		**o**	**0.06**		**u**	**0.07**		**kk u**	**0.02**					**h u**	**0.07**		**p**	**0.05**
	e e	**0.02**		**r**	**0.07**		**e**	**0.06**		**o**	**0.05**		**k u**	**0.02**					**s u**	**0.05**		**b**	**0.05**
	o o	**0.01**		**ch y**	**0.07**		**o o**	**0.05**		**o o**	**0.02**		**h u**	**0.02**					**b u**	**0.04**		**r**	**0.04**
	a a	**0.01**		**p**	**0.06**		**e i**	**0.04**		**i a**	**0.02**		**t o**	**0.01**					**k o**	**0.03**		**d**	**0.04**
				m	**0.06**		**i i**	**0.02**		**e e**	**0.02**		**pp u**	**0.01**					**g a**	**0.03**		**u r**	**0.03**
				ch	**0.06**		**u u**	**0.01**		**e**	**0.02**		**b i**	**0.01**					**s a**	**0.02**		**n y**	**0.03**
AO	**o**	**0.29**	**D**	**d**	**0.16**	**F**	**h**	**0.18**	**JH**	**b**	**0.13**	**NG**	**tt o**	**0.21**	**R**	**r**	**0.53**	**UH**	**a**	**0.24**	**Z**	**t o**	**0.14**
	a	**0.26**		**d o**	**0.15**		**h u**	**0.14**		**k**	**0.1**		**r u**	**0.17**		**n**	**0.07**		**o**	**0.14**		**z u**	**0.11**
	e	**0.14**		**n**	**0.05**		**b**	**0.09**		**j y**	**0.1**		**n**	**0.14**		**u r**	**0.05**		**e**	**0.11**		**r u**	**0.11**
	o o	**0.12**		**t o**	**0.03**		**sh i**	**0.07**		**s**	**0.08**		**kk u**	**0.1**		**r i**	**0.03**		**y u**	**0.1**		**s u**	**0.1**
	i	**0.08**		**sh i**	**0.03**		**p**	**0.07**		**m**	**0.08**		**s u**	**0.07**		**r u**	**0.02**		**a i**	**0.09**		**g u**	**0.09**
	u	**0.05**		**k u**	**0.03**		**m**	**0.06**		**t**	**0.07**		**m u**	**0.06**		**d**	**0.02**		**i**	**0.08**		**m u**	**0.07**
	y u	**0.03**		**k**	**0.03**		**r**	**0.04**		**j**	**0.07**		**dd o**	**0.04**		**t**	**0.01**		**u u**	**0.07**		**n**	**0.06**
	e e	**0.01**		**g u**	**0.03**		**s**	**0.03**		**h**	**0.07**		**tch i**	**0.03**		**s**	**0.01**		**o o**	**0.07**		**d o**	**0.06**
				b	**0.03**		**h a**	**0.03**		**sh**	**0.06**		**pp u**	**0.03**		**m**	**0.01**		**a a**	**0.03**		**j i**	**0.02**
				s	**0.02**		**b u**	**0.02**		**d**	**0.05**		**jj i**	**0.03**		**k**	**0.01**		**u**	**0.02**		**ch i**	**0.02**
AW	**o o**	**0.2**	**DH**	**h**	**0.13**	**G**	**g u**	**0.13**	**K**	**k**	**0.17**	**OW**	**a**	**0.3**	**S**	**s u**	**0.4**	**UW**	**u**	**0.39**	**ZH**	**m**	**0.17**
	a u	**0.19**		**r**	**0.12**		**g**	**0.11**		**n**	**0.1**		**o**	**0.25**		**n**	**0.11**		**a**	**0.15**		**p**	**0.16**
	a	**0.18**		**b**	**0.09**		**k u**	**0.08**		**k u**	**0.1**		**o o**	**0.12**		**r u**	**0.05**		**o**	**0.13**		**t**	**0.15**
	a i	**0.11**		**w**	**0.08**		**b u**	**0.06**		**kk u**	**0.05**		**u**	**0.09**		**t o**	**0.03**		**u u**	**0.12**		**h**	**0.13**
	a a	**0.11**		**t**	**0.07**		**k**	**0.04**		**t o**	**0.03**		**i**	**0.07**		**k u**	**0.03**		**i**	**0.04**		**d**	**0.1**
	e	**0.05**		**p**	**0.07**		**b**	**0.04**		**s u**	**0.03**		**y a**	**0.04**		**sh i**	**0.02**		**y u**	**0.03**		**s**	**0.08**
	o	**0.04**		**g**	**0.06**		**t o**	**0.03**		**sh i**	**0.02**		**e**	**0.04**		**r i**	**0.02**		**i i**	**0.03**		**b**	**0.07**
	i	**0.04**		**j y**	**0.05**		**t**	**0.03**		**r**	**0.02**		**u u**	**0.02**		**m u**	**0.02**		**e**	**0.03**		**r**	**0.05**
	i y	**0.02**		**d**	**0.05**		**h a**	**0.03**		**k o**	**0.02**		**a i**	**0.02**		**h u**	**0.02**		**o o**	**0.02**		**j y**	**0.03**
	e a	**0.01**		**k**	**0.03**		**d**	**0.03**		**k a**	**0.02**		**i i**	**0.01**		**ch i**	**0.02**		**e e**	**0.02**		**k**	**0.02**

Figure 5: Phonemic substitution table learnt from non-parallel corpora. For each English phoneme, only the top ten mappings with P($j|e$) > 0.01 are shown.

model used in decipherment training has a large effect on the learnt P($j|e$) phonemic substitution table (i.e., probabilities for the various phoneme mappings within the WFST C model), which in turn affects the quality of the back-transliterated English output produced when decoding Japanese.

Figure 5 shows the phonemic substitution table learnt using word-based decipherment. The mappings are reasonable, given the lack of parallel data. They are not entirely correct—for example, the mapping "S → s u" is there, but "S → s" is missing.

Sample end-to-end transliterations are illustrated in Figure 6. The figure shows how the transliteration results from non-parallel training improve steadily as we use stronger decipherment techniques. We note that in one case (LAUTENBERG), the decipherment mapping table leads to a correct answer where the mapping table derived from parallel data does not. Because parallel data is limited, it may not contain all of the necessary mappings.

4.4 Size of Japanese Training Data

Monolingual corpora are more easily available than parallel corpora, so we can use increasing amounts of monolingual Japanese training data during decipherment training. The table below shows that using more Japanese training data produces better transliteration results when deciphering with the word-based English model.

Japanese training data *(# of phoneme sequences)*	**Error on name transliteration task** *whole-name error*	*normalized word edit distance*
4,674	87	69.7
9,350	**77**	**57.2**

No Parallel Data Used.

Original	Correct Answer	Parallel Phonetic Training	Decipherment (Method 1)	Decipherment (Method 2)	Decipherment (Method 3)
スペンサー・エーブラハム	**SPENCER ABRAHAM**	**SPENCER ABRAHAM**	**SPENCER** EDELMAN	SPACE CUOMO	**SPENCER ABRAHAM**
ダニエル・アカカ	**DANIEL AKAKA**	**DANIEL** ACA KA	KOREA EAST	SCHERING EAST	DANIELLE ABACO
ウェイン・アラード	**WAYNE ALLARD**	**WAYNE ALLARD**	CHOICE JOHN	**WAYNE** BYRD	**WAYNE ALLARD**
マックス・ボーカス	**MAX BAUCUS**	**MAX BAUCUS**	MEESE JAMES	**MAX** BOOKS	**MAX** FOCUS
ボブ・ベネット	**BOB BENNETT**	**BOB BENNETT**	MY SCHERING	JACK BILLING	BOTH BENNING
ジョセフ・バイデン	**JOSEPH BIDEN**	**JOSEPH BIDEN**	HOUSE LABOR	JAPAN **BIDEN**	**JOSEPH BIDEN**
ジェフ・ビンガマン	**JEFF BINGAMAN**	**JEFF BINGAMAN**	JOHN PFEIFFER	**JEFF** BENJAMIN	**JEFF** BENJAMIN
フランク・ローテンバーク	**FRANK LAUTENBERG**	**FRANK** LAUTENBACH	SUN FLINT	FRANCE **LAUTENBERG**	FRANCE **LAUTENBERG**

Figure 6: Results for end-to-end name transliteration. This figure shows the correct answer, the answer obtained by training mappings on parallel data (Knight and Graehl, 1998), and various answers obtained by deciphering non-parallel data. Method 1 uses a 2-gram P(e), Method 2 uses a 3-gram P(e), and Method 3 uses a word-based P(e).

4.5 P($j|e$) Initialization

So far, the P($j|e$) connections within the WFST C model were initialized with uniform weights prior to EM training. It is a known fact that the EM algorithm does not necessarily find a global minimum for the given objective function. If the search space is bumpy and non-convex as is the case in our problem, EM can get stuck in any of the local minima depending on what weights were used to initialize the search. Different sets of initialization weights can lead to different convergence points during EM training, or in other words, depending on how the P($j|e$) probabilities are initialized, the final P($j|e$) substitution table learnt by EM can vary.

We can use some prior knowledge to initialize the probability weights in our WFST C model, so as to give EM a good starting point to work with. Instead of using uniform weights, in the P($j|e$) model we set higher weights for the mappings where English and Japanese sounds share common consonant phonemes.

For example, mappings such as:

```
N => n        N => a n
D => d        D => d o
```

are weighted X (a constant) times higher than other mappings such as:

```
N  => b       N  => r
D  => B       EY => a a
```

in the P($j|e$) model. In our experiments, we set the value X to 100.

Initializing the WFST C in this way results in EM learning better substitution tables and yields slightly better results for the Senator task. Normalized edit distance drops from 57.2 to 54.2, and the whole-name error is also reduced from 77% to 73% (row 2e in Figure 4).

4.6 Size of English Training Data

We saw earlier (in Section 4.4) that using more monolingual Japanese training data yields improvements in decipherment results. Similarly, we hypothesize that using more monolingual English data can drive the decipherment towards better transliteration results. On the English side, we build different word-based P(e) models, each trained on different amounts of data (English phoneme sequences from the CMU dictionary). The table below shows that deciphering with a word-based English model

built from more data produces better transliteration results.

English training data *(# of phoneme sequences)*	**Error on name transliteration task** *whole-name error*	*normalized word edit distance*
76,152	73	54.2
97,912	**66**	**49.3**

This yields the best transliteration results on the Senator task with non-parallel data, getting 34 out of 100 Senator names exactly right.

4.7 Re-ranking Results Using the Web

It is possible to improve our results on the U.S. Senator task further using external monolingual resources. Web counts are frequently used to automatically re-rank candidate lists for various NLP tasks (Al-Onaizan and Knight, 2002). We extract the top 10 English candidates produced by our word-based decipherment method for each Japanese test name. Using a search engine, we query the entire English name (first and last name) corresponding to each candidate, and collect search result counts. We then re-rank the candidates using the collected Web counts and pick the most frequent candidate as our choice.

For example, *France Murkowski* gets only 1 hit on Google, whereas *Frank Murkowski* gets 135,000 hits. Re-ranking the results in this manner lowers the whole-name error on the Senator task from 66% to 61%, and also lowers the normalized edit distance from 49.3 to 48.8. However, we do note that re-ranking using Web counts produces similar improvements in the case of parallel training as well and lowers the whole-name error from 40% to 24%.

So, the re-ranking idea, which is simple and requires only monolingual resources, seems like a nice strategy to apply at the end of transliteration experiments (during decoding), and can result in further gains on the final transliteration performance.

5 Comparable versus Non-Parallel Corpora

We also present decipherment results when using comparable corpora for training the WFST C model. We use English and Japanese phoneme sequences derived from a parallel corpus containing 2,683 phoneme sequence pairs to construct comparable corpora (such that for each Japanese phoneme sequence, the correct back-transliterated phoneme sequence is present somewhere in the English data) and apply the same decipherment strategy using a word-based English model. The table below compares the transliteration results for the U.S. Senator task, when using comparable versus non-parallel data for decipherment training. While training on comparable corpora does have benefits and reduces the whole-name error to 59% on the Senator task, it is encouraging to see that our best decipherment results using only non-parallel data comes close (66% error).

English/Japanese Corpora *(# of phoneme sequences)*	**Error on name transliteration task** *whole-name error*	*normalized word edit distance*
Comparable Corpora *(English = 2,608 Japanese = 2,455)*	59	41.8
Non-Parallel Corpora *(English = 98,000 Japanese = 9,350)*	66	49.3

6 Conclusion

We have presented a method for attacking machine transliteration problems without parallel data. We developed phonemic substitution tables trained using only monolingual resources and demonstrated their performance in an end-to-end name transliteration task. We showed that consistent improvements in transliteration performance are possible with the use of strong decipherment techniques, and our best system achieves significant improvements over the baseline system. In future work, we would like to develop more powerful decipherment models and techniques, and we would like to harness the information available from a wide variety of monolingual resources, and use it to further narrow the gap between parallel-trained and non-parallel-trained approaches.

7 Acknowledgements

This research was supported by the Defense Advanced Research Projects Agency under SRI International's prime Contract Number NBCHD040058.

References

Y. Al-Onaizan and K. Knight. 2002. Translating named entities using monolingual and bilingual resources. In *Proc. of ACL*.

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 Series*, 39(4):1–38.

Y. Goldberg and M. Elhadad. 2008. Identification of transliterated foreign words in Hebrew script. In *Proc. of CICLing*.

D. Goldwasser and D. Roth. 2008a. Active sample selection for named entity transliteration. In *Proc. of ACL/HLT Short Papers*.

D. Goldwasser and D. Roth. 2008b. Transliteration as constrained optimization. In *Proc. of EMNLP*.

S. Goldwater and L. Griffiths, T. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proc. of ACL*.

J. Graehl. 1997. Carmel finite-state toolkit. http://www.isi.edu/licensed-sw/carmel.

L. Haizhou, Z. Min, and S. Jian. 2004. A joint source-channel model for machine transliteration. In *Proc. of ACL*.

U. Hermjakob, K. Knight, and H. Daume. 2008. Name translation in statistical machine translation—learning when to transliterate. In *Proc. of ACL/HLT*.

F. Huang, S. Vogel, and A. Waibel. 2004. Improving named entity translation combining phonetic and semantic similarities. In *Proc. of HLT/NAACL*.

S. Karimi, F. Scholer, and A. Turpin. 2007. Collapsed consonant and vowel models: New approaches for English-Persian transliteration and back-transliteration. In *Proc. of ACL*.

A. Klementiev and D. Roth. 2008. Named entity transliteration and discovery in multilingual corpora. In *Learning Machine Translation*. MIT press.

K. Knight and J. Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

K. Knight and K. Yamada. 1999. A computational approach to deciphering unknown scripts. In *Proc. of the ACL Workshop on Unsupervised Learning in Natural Language Processing*.

K. Knight, A. Nair, N. Rathod, and K. Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proc. of COLING/ACL*.

J. Kuo, H. Li, and Y. Yang. 2006. Learning transliteration lexicons from the web. In *Proc. of ACL/COLING*.

H. Li, C. Sim, K., J. Kuo, and M. Dong. 2007. Semantic transliteration of personal names. In *Proc. of ACL*.

M. Nagata, T. Saito, and K. Suzuki. 2001. Using the web as a bilingual dictionary. In *Proc. of the ACL Workshop on Data-driven Methods in Machine Translation*.

J. Oh and H. Isahara. 2006. Mining the web for transliteration lexicons: Joint-validation approach. In *Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence*.

S. Ravi and K. Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *Proc. of EMNLP*.

S. Ravi and K. Knight. 2009. Probabilistic methods for a Japanese syllable cipher. In *Proc. of the International Conference on the Computer Processing of Oriental Languages (ICCPOL)*.

T. Sherif and G. Kondrak. 2007a. Bootstrapping a stochastic transducer for arabic-english transliteration extraction. In *Proc. of ACL*.

T. Sherif and G. Kondrak. 2007b. Substring-based transliteration. In *Proc. of ACL*.

R. Sproat, T. Tao, and C. Zhai. 2006. Named entity transliteration with comparable corpora. In *Proc. of ACL*.

T. Tao, S. Yoon, A. Fister, R. Sproat, and C. Zhai. 2006. Unsupervised named entity transliteration using temporal and phonetic correlation. In *Proc. of EMNLP*.

J. Wu and S. Chang, J. 2007. Learning to find English to Chinese transliterations on the web. In *Proc. of EMNLP/CoNLL*.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*.

S. Yoon, K. Kim, and R. Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *Proc. of ACL*.

D. Zelenko and C. Aone. 2006. Discriminative methods for transliteration. In *Proc. of EMNLP*.

A Corpus-Based Approach for the Prediction of Language Impairment in Monolingual English and Spanish-English Bilingual Children

Keyur Gabani and **Melissa Sherman** and **Thamar Solorio** and **Yang Liu**
Department of Computer Science
The University of Texas at Dallas
keyur,mesh,tsolorio,yangl@hlt.utdallas.edu

Lisa M. Bedore and **Elizabeth D. Peña**
Department of Communication Sciences and Disorders
The University of Texas at Austin
lbedore,lizp@mail.utexas.edu

Abstract

In this paper we explore a learning-based approach to the problem of predicting language impairment in children. We analyzed spontaneous narratives of children and extracted features measuring different aspects of language including morphology, speech fluency, language productivity and vocabulary. Then, we evaluated a learning-based approach and compared its predictive accuracy against a method based on language models. Empirical results on monolingual English-speaking children and bilingual Spanish-English speaking children show the learning-based approach is a promising direction for automatic language assessment.

1 Introduction

The question of how best to identify children with language disorders is a topic of ongoing debate. One common assessment approach is based on cutoff scores from standardized, norm-referenced language assessment tasks. Children scoring at the lower end of the distribution, typically more than 1.25 or 1.5 Standard Deviations (SD) below the mean, are identified as having language impairment (Tomblin et al., 1997). This cutoff-based approach has several well-documented weaknesses that may result in both over- and under-identification of children as language impaired (Plante and Vance, 1994). Recent studies have suggested considerable overlap between children with language impairment and their typically developing cohorts on many of these tasks (e.g., (Peña et al., 2006b; Spaulding et al., 2006)). In addition, scores and cutoffs on standardized tests depend on the distribution of scores from the particular samples used in normalizing the measure. Thus, the validity of the measure for children whose demographic and other socioeconomic characteristics are not well represented in the test's normative sample is a serious concern. Finally, most norm-referenced tests of language ability rely heavily on exposure to mainstream language and experiences, and have been found to be biased against children from families with low parental education and socioeconomic status, as well as children from different ethnic backgrounds (Campbell et al., 1997; Dollaghan and Campbell, 1998).

This paper aims to develop a reliable and automatic method for identifying the language status of children. We propose the use of different lexico-syntactic features, typically used in computational linguistics, in combination with features inspired by current assessment practices in the field of language disorders to train Machine Learning (ML) algorithms. The two main contributions of this paper are: 1) It is one step towards developing a reliable and automatic approach for language status prediction in English-speaking children; 2) It provides evidence showing that the same approach can be adapted to predict language status in Spanish-English bilingual children.

2 Related Work

2.1 Monolingual English-Speaking Children

Several hypotheses exist that try to explain the grammatical deficits of children with Language Impair-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 46–55,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

ment (LI). Young children normally go through a stage where they use non-finite forms of verbs in grammatical contexts where finite forms are required (Wexler, 1994). This is referred as the optional infinitive stage. The Extended Optional Infinitive (EOI) theory (Rice and Wexler, 1996) suggests that children with LI exhibit the use of a "young" grammar for an extended period of time, where tense, person, and number agreement markers are omitted.

In contrast to the EOI theory, the surface account theory (Leonard et al., 1997) assumes that children with LI have reduced processing capabilities. This deficit affects the perception of low stress morphemes, such as *-ed, -s, be* and *do*, resulting in an inconsistent use of these verb morphemes.

Spontaneous narratives are considered as one of the most ecologically valid ways to measure communicative competence (Botting, 2002). They represent various aspects involved in children's everyday communication. Typical measures for spontaneous language samples include Mean Length of Utterance (MLU) in words, Number of Different Words (NDW), and errors in grammatical morphology. Assessment approaches compare children's performance on these measures against expected performance. As mentioned in Section 1, these cut-off based methods raise questions concerning accuracy and bias. Manually analyzing the narratives is also a very time consuming task. After transcribing the sample, clinicians need to code for the different clinical markers and other morphosyntactic information. This can take up to several hours for each child making it infeasible to analyze a large number of samples.

2.2 Bilingual Spanish-English Speaking Children

Bilingual children face even more identification challenges due to their dual language acquisition. They can be mistakenly labeled as LI due to: 1) the inadequate use of translations of assessment tools; 2) an over reliance on features specific to English; 3) a lack of appropriate expectations about how the languages of a bilingual child should develop (Bedore and Peña, 2008); 4) or the use of standardized tests where the normal distribution used to compare language performance is composed of monolingual children (Restrepo and Gutiérrez-Clellen, 2001).

Spanish speaking children with LI show different clinical markers than English speaking children with LI. As mentioned above, English speakers have problems with verb morphology. In contrast, Spanish speakers have been found to have problems with noun morphology, in particular in the use of articles and clitics (Restrepo and Gutiérrez-Clellen, 2001; Jacobson and Schwartz, 2002; Bedore and Leonard, 2005). Bedore and Leonard (2005) also found differences in the error patterns of Spanish and related languages such as Italian. Spanish-speakers tend to both omit and substitute articles and clitics, while the dominant errors for Italian-speakers are omissions.

3 Our Approach

We use language models (LMs) in our initial investigation, and later explore more complex ML algorithms to improve the results. Our ultimate goal is to discover a highly accurate ML method that can be used to assist clinicians in the task of LI identification in children.

3.1 Language Models for Predicting Language Impairment

LMs are statistical models used to estimate the probability of a given sequence of words. They have been explored previously for clinical purposes. Roark *et al.* (2007) proposed cross entropy of LMs trained on Part-of-Speech (POS) sequences as a measure of syntactic complexity with the aim of determining mild cognitive impairment in adults. Solorio and Liu (2008) evaluated LMs on a small data set in a preliminary trial on LI prediction.

The intuition behind using LMs is that they can identify atypical grammatical patterns and help discriminate the population with potential LI from the Typically Developing (TD) one. We use LMs trained on POS tags rather than on words. Using POS tags can address the data sparsity issue in LMs, and place less emphasis on the vocabulary and more emphasis on the syntactic patterns.

We trained two separate LMs using POS tags from the transcripts of TD and LI children, respectively. The language status of a child is predicted using the following criterion:

$$d(s) = \begin{cases} \text{LI} & if\ (PP_{TD}(s) > PP_{LI}(s)) \\ \text{TD} & otherwise \end{cases}$$

where s represents a transcript from a child, and $PP_{TD}(s)$ and $PP_{LI}(s)$ are the perplexity values from the TD and LI LMs, respectively. We used the SRI Language Modeling Toolkit (Stolcke, 2002) for training the LMs and calculating perplexities.

3.2 Machine Learning for Predicting Language Impairment

Although LMs have been used successfully on different human language processing tasks, they are typically trained and tested on language samples larger than what is usually collected by clinicians for the purpose of diagnosing a child with potential LI. Clinicians make use of additional information beyond children's speech, such as parent and teacher questionnaires and test scores on different language assessment tasks. Therefore in addition to using LMs for children language status prediction, we explore a machine learning classification approach that can incorporate more information for better prediction. We aim to identify effective features for this task and expect this information will help clinicians in their assessment.

We consider various ML algorithms for the classification task, including Naive Bayes, Artificial Neural Networks (ANNs), Support Vector Machines (SVM), and Boosting with Decision Stumps. Weka (Witten and Frank, 1999) was used in our experiments due to its known reliability and the availability of a large number of algorithms. Below we provide a comprehensive list of features that we explored for both English and Spanish-English transcripts. We group these features according to the aspect of language they focus on. Features specific to Spanish are discussed in Section 5.2.

1. *Language productivity*

   (a) *Mean Length of Utterance (MLU) in words*

   Due to a general deficit of language ability, children with LI have been found to produce language samples with a shorter MLU in words because they produce grammatically simpler sentences when compared to their TD peers.

   (b) *Total number of words*

   This measure is widely used when building language profiles of children for diagnostic and treatment purposes.

   (c) *Degree of support*

   In spontaneous samples of children's speech, it has been pointed out that children with potential LI need more encouragement from the investigator (Wetherell et al., 2007) than their TD peers. A support prompt can be a question like "*What happened next*?" We count the number of utterances, or turns, of the investigator interviewing the child for this feature.

2. *Morphosyntactic skills*

   (a) *Ratio of number of raw verbs to the total number of verbs*

   As mentioned previously, children with LI omit tense markers in verbs more often than their TD cohorts. For example:

   *...the boy **look** into the hole but didn't find...*

   Hence, we include the ratio of the number of raw verbs to the total number of verbs as a feature.

   (b) *Subject-verb agreement*

   Research has shown that English-speaking children with LI have difficulties marking subject-verb agreement (Clahsen and Hansen, 1997; Schütze and Wexler, 1996). An illustration of subject-verb disagreement is the following:

   *...and **he were** looking behind the rocks*

   As a way of capturing this information in the machine learning setting, we consider various bigrams of POS tags: noun and verb, noun and auxiliary verb, pronoun and verb, and pronoun and auxiliary verb. These features are included in a bag-of-words fashion using individual counts. Also, we allow a window between these pairs to capture agreement between sub-

ject and verb that may have modifiers in between.

(c) *Number of different POS tags*
This feature is the total number of different POS tags in each transcript.

3. *Vocabulary knowledge*

We use the Number of Different Words (NDW) to represent vocabulary knowledge of a child. Although such measures can be biased against children from different backgrounds, we expect this possible negative effect to decrease as a result of having a richer pool of features.

4. *Speech fluency*

Repetitions, revisions, and filled pauses have been considered indicators of language learning difficulties (Thordardottir and Weismer, 2002; Wetherell et al., 2007). In this work we include as features (a) the number of fillers, such as *uh, um, er*; and (b) the number of disfluencies (abandoned words) found in each transcript.

5. *Perplexities from LMs*

As mentioned in Section 3.1 we trained LMs of order 1, 2, and 3 on POS tags extracted from TD and LI children. We use the perplexity values from these models as features. Additionally, differences in perplexity values from LI and TD LMs for different orders are used as features.

6. *Standard scores*

A standard score, known as a z-score, is the difference between an observation and the mean relative to the standard deviation. For this feature group, we first find separate distributions for the MLU in words, NDW and total number of utterances for the TD and LI populations. Then, for each transcript, we compute the standard scores based on each of these six distributions. This represents how well the child is performing relative to the TD and LI populations. Note that a cross validation setup was used to obtain the distribution for the TD and LI children for training. This is also required for the LM features above.

4 Experiments with Monolingual Children

4.1 The Monolingual English Data Set

Our target population for this work is children with an age range of 3 to 6 years old. However, currently we do not have any monolingual data sets readily available to test our approach in this age range. In the field of communication disorders data sharing is not a common practice due to the sensitive content of the material in the language samples of children, and also due to the large amount of effort and time it takes researchers to collect, transcribe, and code the data before they can begin their analysis. To evaluate our approach we used a dataset from CHILDES (MacWhinney, 2000) that includes narratives from English-speaking adolescents with and without LI with ages ranging between 13 and 16 years old. Even though the age range is outside the range we are interested in, we believe that this data set can still be helpful in exploring the feasibility of our approach as a first step.

This data set contains 99 TD adolescents and 19 adolescents who met the LI profile at one point in the duration of the study. There are transcripts from each child for two tasks: a story telling and a spontaneous personal narrative. The first task is a picture prompted story telling task using the wordless picture book, "*Frog, Where Are You?*" (Mayer, 1969). In this story telling task children first look at the story book –to develop a story in memory– and then are asked to narrate the story. This type of elicitation task encourages the use of past tense constructions, providing plenty of opportunities for extracting clinical markers. In the spontaneous personal narrative task, the child is asked to talk about a person who annoys him/her the most and describe the most annoying features of that person. This kind of spontaneous personal narrative encourages the participant for the use of third person singular forms (*-s*). Detailed information of this data set can be found in (Wetherell et al., 2007).

We processed the transcripts using the CLAN toolkit (MacWhinney, 2000). MOR and POST from CLAN are used for morphological analysis and POS tagging of the children's speech. We decided to use these analyzers since they are customized for children's speech.

	Story telling			Personal narrative		
Method	**P (%)**	**R (%)**	**F_1 (%)**	**P (%)**	**R (%)**	**F_1 (%)**
Baseline	28.57	10.53	15.38	33.33	15.79	21.43
1-gram LMs	41.03	84.21	55.17	34.21	68.42	45.61
2-gram LMs	75.00	47.37	**58.06**	55.56	26.32	35.71
3-gram LMs	80.00	21.05	33.33	87.50	36.84	**51.85**

Table 1: Evaluation of language models on the monolingual English data set.

	Story telling			Personal narrative		
Algorithm	**P (%)**	**R (%)**	**F_1 (%)**	**P (%)**	**R (%)**	**F_1 (%)**
Naive Bayes	38.71	63.16	48.00	34.78	42.11	38.10
Bayesian Network	58.33	73.68	65.12	28.57	42.11	34.04
SVM	76.47	68.42	**72.22**	47.06	42.11	44.44
ANNs	62.50	52.63	57.14	50.00	47.37	48.65
Boosting	70.59	63.16	66.67	69.23	47.37	**56.25**

Table 2: Evaluation of machine learning algorithms on the monolingual English data set.

4.2 Results with Monolingual English-Speaking Children

The performance measures we use are: precision (P), recall (R), and F-measure (F_1). Here the LI category is the positive class and the TD category is the negative class.

Table 1 shows the results of leave-one-out-cross-validation (LOOCV) obtained from the LM approach for the story telling and spontaneous personal narrative tasks. It also shows results from a baseline method that predicts language status by using standard scores on measures that have been associated with LI in children (Dollaghan, 2004). The three measures we used for the baseline are: MLU in words, NDW, and total number of utterances produced. To compute this baseline we estimate the mean and standard deviation of these measures using LOOCV with the TD population as our normative sample. The baseline predicts that a child has LI if the child scores more than 1.25 SD below the mean on at least two out of the three measures.

Although LMs yield different results for the story telling and personal narrative tasks, they both provide consistently better results than the baseline. For the story telling task the best results, in terms of the F_1 measure, are achieved by a bigram LM ($F_1 = 58.06\%$) while for the personal narrative the highest F_1 measure (51.85%) is from the trigram LM. If we consider precision, both tasks have the same increasing pattern when increasing LM orders. However for recall that is not the case. In the story telling task, recall decreases at the expense of higher precision, but for the personal narrative task, the trigram LM reaches a better trade-off between precision and recall, which yields a high F_1 measure. We also evaluated 4-gram LMs, but results did not improve, most likely because we do not have enough data to train higher order LMs.

The results for different ML algorithms are shown in Table 2, obtained by using all features described in Section 3.2. The feature based approach using ML algorithms outperformed using only LMs on both tasks. For the story telling task, SVM with a linear kernel achieves the best results ($F_1 = 72.22\%$), while Boosting with Decision Stumps provides the best performance ($F_1 = 56.25\%$) for the personal narrative task.

4.3 Feature and Error Analysis

The ML results shown above use the entire feature set described in Subsection 3.2. The next question we ask is the effectiveness of different features for this task. The datasets we are using in our evaluation are very small, especially considering the number of positive instances. This prevents us from having a separate subset of the data for parameter tuning or feature selection. Therefore, we performed additional experiments to evaluate the usefulness of individual features. Figure 1 shows the F_1 measures

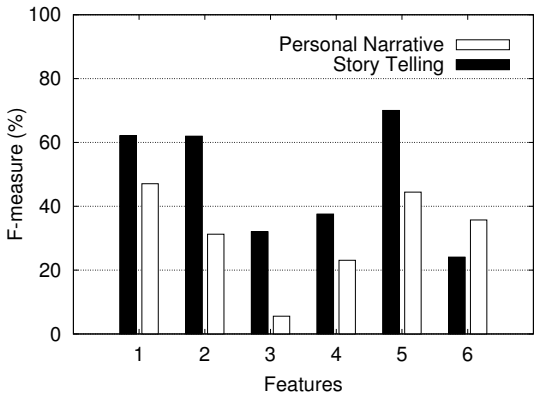


Figure 1: Discriminating power of different groups of features. The numbers on the x-axis correspond to the feature groups in Section 3.2.

when using different feature groups. The numbers on the x-axis correspond to the feature groups described in Section 3.2. The F_1 measure value for each of the features is the highest value obtained by running different ML algorithms for classification.

We noticed that for the story telling task, using perplexity values from LMs (group 5) as a feature in the ML setting outperforms the LM threshold approach by a large margin. It seems that having the perplexity values as well as the perplexity differences from all the LMs of different orders in the ML algorithm provides a better estimation of the target concept.

Only the standard scores (group 6) yield a higher F_1 measure for the personal narrative task than the story telling one. The majority of the features (5 out of 6 groups) provide higher F_1 measures for the story telling task, which explains the significantly better results on this task over the personal narrative in our learning approach. This is consistent with previous work contrasting narrative genre stating that the restrictive setting of a story retell is more revealing of language difficulties than spontaneous narratives, where the subjects have more control on the content and style (Wetherell et al., 2007).

We also performed some error analysis for some of the transcripts that were consistently misidentified by different ML algorithms. In the story telling task, we find that some LI transcripts are misclassified as TD because they (1) have fewer fillers, disfluencies, and degree of support; (2) are similar to the TD transcripts, which is depicted by the perplexity values for these transcripts; or (3) contain higher MLU in words as compared to their LI peers. Some of the reasons for classifying transcripts in the TD category as LI are shorter MLU in words as compared to other TD peers, large number of fillers, and excessive repetitions of words and phrases unlike the other TD children. These factors are consistent with the effective features that we found from Figure 1.

For the personal narrative task, standard scores (group 6) and language productivity (group 1) have an important role in classification, as shown in Figure 1. The TD transcripts that are misidentified have lower standard scores and MLU in words than those of their TD peers.

We believe that another source of noise in the transcripts comes from the POS tags themselves. For instance, we found that many verbs in present tense for third person singular are tagged as plural nouns, which results in a failure to capture subject-verb agreement.

Lastly, according to the dataset description, children in the LI category met the LI criteria at one stage in their lifetime and some of these children also had, or were receiving, some educational support in the school environment at the time of data collection. This support for children with LI is meant to improve their performance on language related tasks, making the automatic classification problem more complicated. This also raises the question about the reference label (TD or LI) for each child in the data set we used. The details about which children received interventions are not specified in the dataset description.

5 Experiments with Bilingual Children

In this section we generalize the approach to a Spanish-English bilingual population. In adapting the approach to our bilingual population we face two challenges: first, what shows to be promising for a monolingual and highly heterogeneous population may not be as successful in a bilingual setting where we expect to have a large variability of exposure to each language; second, there is a large difference in the mean age of the monolingual setting and that of our bilingual one. This age difference will result in different speech patterns. Younger children pro-

duce more ill-formed sentences since they are still in a language acquisition phase. Lastly, the clinical markers in adolescents are geared towards problems at the pragmatic and discourse levels, while at younger ages they focus more on syntax and morphology.

For dealing with the first challenge we are extracting language-specific features and hope that by looking at both languages we can reach a good discrimination performance. For the second challenge, our feature engineering approach has been focused on younger children from the beginning. We are aiming to capture the type of morphosyntactic patterns that can identify LI in young children. In addition, the samples in the bilingual population are story retells, and our feature setting showed to be a good match for this task. Therefore, we expect our approach to capture relevant classification patterns, even in the presence of noisy utterances.

5.1 The Bilingual Data Set

The transcripts for the bilingual LI task come from an on-going longitudinal study of language impairment in Spanish-English speaking children (Peña et al., 2006a). The children in this study were enrolled in kindergarten with a mean age of about 70 months. Of the 59 children, 6 were identified as having a possible LI by an expert in communication disorders, while 53 were identified as TD. Six of the TD children were excluded due to missing information, yielding a total of 47 TD children.

Each child told a series of stories based on Mercer Mayer's wordless picture books (Mayer, 1969). Two stories were told in English and two were told in Spanish, for a total of four transcripts per child. The books used for English were *"A Boy, A Dog, and A Frog"* and *"Frog, Where Are You?"* The books used for Spanish retelling were *"Frog on His Own"* and *"Frog Goes to Dinner."* The transcripts for each separate language were combined, yielding one instance per language for each child.

An interesting aspect of the bilingual data is that the children mix languages in their narratives. This phenomenon is called code-switching. At the beginning of a retelling session, the interviewer encourages the child to speak the target language if he/she is not doing so. Once the child begins speaking the correct language, any code-switching thereafter is not corrected by the interviewer. Due to this, the English transcripts contain Spanish utterances and vice versa. We believe that words in the non-target language help contribute to a more accurate language development profile. Therefore, in our work we decided to keep these code-switched elements. A combined lexicon approach was used to tag the mixed-language fragments. If a word does not appear in the target language lexicon, we apply the POS tag from the non-target language.

5.2 Spanish-Specific Features

Many structural differences exist between Spanish, a Romance language, and English, a Germanic language. Spanish is morphologically richer than English. It contains a larger number of different verb conjugations and it uses a two gender system for nouns, adjectives, determiners, and participles. A Spanish-speaking child with LI will have difficulties with different grammatical elements, such as articles and clitics, than an English-speaking child (Bedore and Peña, 2008). These differences indicate that the Spanish feature set will need to be tailored towards the Spanish language.

To account for Spanish-specific patterns we included new POS bigrams as features. To capture the use of correct and incorrect gender and number marking morphology, we added noun-adjective, determiner-noun, and number-noun bigrams to the list of morphosyntactic features.

5.3 Results on Bilingual Children

Results are shown for the baseline and LM threshold approach for the bilingual data set in Table 3. The baseline is computed from the same measures as the monolingual dataset (MLU in words, NDW, and total utterances).

Compared to Table 1, the values in Table 3 are generally lower than on the monolingual story telling task. In this inherently difficult task, the bilingual transcripts are more disfluent than the monolingual ones. This could be due to the age of the children or their bilingual status. Recent studies on psycholinguistics and language production have shown that bilingual speakers have both languages active at speech production time (Kroll et al., 2008) and it is possible that this may cause interference, especially in children still in the phase of language acqui-

	English			Spanish		
Method	**P (%)**	**R (%)**	**F_1 (%)**	**P (%)**	**R (%)**	**F_1 (%)**
Baseline	20.00	16.66	18.18	16.66	16.66	16.66
1-gram LMs	40.00	33.33	36.36	17.64	50.00	**26.08**
2-gram LMs	50.00	33.33	40.00	33.33	16.66	22.22
3-gram LMs	100.00	33.33	**50.00**	0.00	0.00	-

Table 3: Evaluation of language models on Bilingual Spanish-English data set.

sition. In addition, the LMs in the monolingual task were trained using more instances per class, possibly yielding better results.

There are some different patterns between using the English and Spanish transcripts. In English, the unigram models provide the least discriminative value, and the bigram and trigram models improve discrimination. We also evaluated higher order n-grams, but did not obtain any further improvement. We found that the classification accuracy of the LM approach was influenced by two children with LI who were consistently marked as LI due to a greater perplexity value from the TD LM. A further analysis shows that these children spoke mostly Spanish on the "English" tasks yielding larger perplexities from the TD LM, which was trained from mostly English. In contrast, the LI LM was created with transcripts containing more Spanish than the TD one, and thus test transcripts with a lot of Spanish do not inflate perplexity values that much.

For Spanish, unigram LMs provide some discriminative usefulness, and then the bigram performance decreases while the trigram model provides no discriminative value. One reason for this may be that the Spanish LMs have a larger vocabulary. In the Spanish LMs, there are 2/3 more POS tags than in the English LM. This size difference dramatically increases the possible bigrams and trigrams, therefore increasing the number of parameters to estimate. In addition, we are using an "off the shelf" POS tagger (provided by CLAN) and this may add noise in the feature extraction process. Since we do not have gold standard annotations for these transcripts, we cannot measure the POS tagging accuracy. A rough estimate based on manually revising one transcript in each language showed a POS tagging accuracy of 90% for English and 84% for Spanish. Most of the POS tagger errors involve verbs, nouns and pronouns. Thus while the accuracy might not seem that low, it can still have a major impact on our approach since it involves the POS categories that are more relevant for this task.

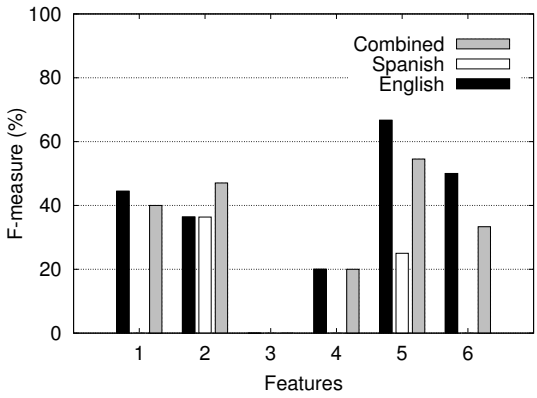


Figure 2: Discriminating power of different groups of features for the bilingual population. The numbers on the x-axis correspond to the feature groups in Section 3.2.

Table 4 shows the results from various ML algorithms. In addition to predicting the language status with the English and Spanish samples separately, we also combined the English and Spanish transcripts together for each child, and used all the features from both languages in order to allow a prediction based on both samples. The best F_1 measure for this task (60%) is achieved by using the Naive Bayes algorithm with the combined Spanish-English feature set. This is an improvement over both the separate English and Spanish trials. The Naive Bayes algorithm provided the best discrimination for the English (54%) and Combined data sets and Boosting and SVM provided the best discrimination for the Spanish set (18%).

5.4 Feature Analysis

Similar to the monolingual dataset, we performed additional experiments exploring the contribution of different groups of features. We tested the six

	English			Spanish			Combined		
Algorithm	**P (%)**	**R (%)**	**F_1 (%)**	**P (%)**	**R (%)**	**F_1 (%)**	**P (%)**	**R (%)**	**F_1 (%)**
ANNs	66.66	33.33	44.44	0.00	0.00	-	100.00	16.66	28.57
SVM	14.28	16.66	15.38	20.00	16.66	18.18	66.66	33.33	44.44
Naive Bayes	60.00	50.00	**54.54**	0.00	0.00	-	75.00	50.00	**60.00**
Logistic Regression	25.00	16.66	20.00	-	0.00	-	50.00	33.33	40.00
Boosting	50.00	33.33	40.00	20.00	16.66	18.18	66.66	33.33	44.44

Table 4: Evaluation of machine learning algorithms on the Bilingual Spanish-English data set.

groups of features described in Section 3.2 separately. Overall, the combined LM perplexity values (group 5) provided the best discriminative value ($F_1 = 66\%$). The LM perplexity values performed the best for English. It even outperformed using all the features in the ML algorithm, suggesting some feature selection is needed for this task.

The morpohsyntactic skills (group 2) provided the best discriminative value for the Spanish language features, and performed better than the complete feature set for Spanish. Within group 2, we evaluated different POS bigrams for the Spanish and English sets and observed that most of the bigram combinations by themselves are usually weak predictors of language status. In the Spanish set, out of all of the lexical combinations, only the determiner-noun, noun-verb, and pronoun-verb categories provided some discriminative value. The determiner-noun category captured the correct and incorrect gender marking between the two POS tags. The noun-verb and pronoun-verb categories covered the correct and incorrect usage of subject-verb combinations. Interestingly enough, the pronoun-verb category performed well by itself, yielding an F_1 measure of 54%. There are also some differences in the frequencies of bigram features in the English and Spanish data sets. For example, there is no noun-auxiliary POS pattern in Spanish, and the pronoun-auxiliary bigram appears less frequently in Spanish than in English because in Spanish the use of personal pronouns is not mandatory since the verb inflection will disambiguate the subject of the sentence.

The vocabulary knowledge feature (group 3) did not provide any discriminative value for any of the language tasks. This may be because bilingual children receive less input for each language than a monolingual child learning one language, or due to the varied vocabulary acquisition rate in our bilingual population.

6 Conclusions and Future Work

In this paper we present results on the use of LMs and ML techniques trained on features representing different aspects of language gathered from spontaneous speech samples for the task of assisting clinicians in determining language status in children. First, we evaluate our approach on a monolingual English-speaking population. Next, we show that this ML approach can be successfully adapted to a bilingual Spanish-English population. ML algorithms provide greater discriminative power than only using a threshold approach with LMs.

Our current efforts are devoted to improving prediction accuracy by refining our feature set. We are working on creating a gold standard corpus of children's transcripts annotated with POS tags. This data set will help us improve accuracy on our POS-based features. We are also exploring the use of socio-demographic features such as the educational level of parents, the gender of children, and enrollment status on free lunch programs.

Acknowledgments

This work was supported by NSF grant 0812134, and by grant 5 UL1 RR024982 from NCRR, a component of NIH. We also thank the three NAACL reviewers for insightful comments on the submitted version of this paper.

References

Lisa M. Bedore and Laurence B. Leonard. 2005. Verb inflections and noun phrase morphology in the spontaneous speech of Spanish-speaking children with specific language impairment. *Applied Psycholinguistics*, 26(2):195–225.

Lisa M. Bedore and Elizabeth D. Peña. 2008. Assessment of bilingual children for identification of language impairment: Current findings and implications for practice. *International Journal of Bilingual Education and Bilingualism*, 11(1):1–29.

Nicola Botting. 2002. Narrative as a tool for the assessment of linguistic and pragmatic impairments. *Child Language Teaching and Therapy*, 18(1):1–21.

Thomas Campbell, Chris Dollaghan, Herbert Needleman, and Janine Janosky. 1997. Reducing bias in language assessment: Processing-dependent measures. *Journal of Speech, Language, and Hearing Research*, 40(3):519–525.

Harald Clahsen and Detlef Hansen. 1997. The grammatical agreement deficit in specific language impairment: Evidence from therapy experiments. In Myrna Gopnik, editor, *The Inheritance and Innateness of Grammar*, chapter 7. Oxford University Press, New York.

Christine A. Dollaghan and Thomas F. Campbell. 1998. Nonword repetition and child language impairment. *Journal of Speech, Language, and Hearing Research*, 41(5):1136–1146.

Christine A. Dollaghan. 2004. Taxometric analyses of specific language impairment in 3- and 4-year-old children. *Journal of Speech, Language, and Hearing Research*, 47(2):464–475.

Peggy F. Jacobson and Richard G. Schwartz. 2002. Morphology in incipient bilingual Spanish-speaking preschool children with specific language impairment. *Applied Psycholinguistics*, 23(1):23–41.

Judith F. Kroll, Chip Gerfen, and Paola E. Dussias. 2008. Laboratory designs and paradigms: Words, sounds, sentences. In L. Wei and M. G. Moyer, editors, *The Blackwell Guide to Research Methods in Bilingualism and Multilingualism*, chapter 7. Blackwell Pub.

Laurence B. Leonard, Julia A. Eyer, Lisa M. Bedore, and Bernard G. Grela. 1997. Three accounts of the grammatical morpheme difficulties of English-speaking children with specific language impairment. *Journal of Speech, Language, and Hearing Research*, 40(4):741–753.

Brian MacWhinney. 2000. *The CHILDES project: Tools for analyzing talk*. Lawrence Erlbaum, Mahwah, NJ.

Mercer Mayer. 1969. *Frog, where are you?* Dial Press.

Elizabeth D. Peña, Lisa M. Bedore, Ronald B. Gillam, and Thomas Bohman. 2006a. Diagnostic markers of language impairment in bilingual children. Grant awarded by the NIDCD, NIH.

Elizabeth D. Peña, Tammie J. Spaulding, and Elena Plante. 2006b. The composition of normative groups and diagnostic decision making: Shooting ourselves in the foot. *American Journal of Speech-Language Pathology*, 15(3):247–254.

Elena Plante and Rebecca Vance. 1994. Selection of preschool language tests: A data-based approach. *Language, Speech, and Hearing Services in Schools*, 25(1):15–24.

María Adelaida Restrepo and Vera F. Gutiérrez-Clellen. 2001. Article use in Spanish-speaking children with specific language impairment. *Journal of Child Language*, 28(2):433–452.

Mabel L. Rice and Kenneth Wexler. 1996. Toward tense as a clinical marker of specific language impairment in English-speaking children. *Journal of Speech and Hearing Research*, 39(6):1239–1257.

Brian Roark, Margaret Mitchell, and Kristy Hollingshead. 2007. Syntactic complexity measures for detecting mild cognitive impairment. In *Proceedings of the Workshop on BioNLP 2007*, pages 1–8. ACL.

Carson T. Schütze and Kenneth Wexler. 1996. Subject case licensing and English root infinitives. In *Proceedings of the 20th Annual Boston University Conference on Language Development*. Cascadilla Press.

Thamar Solorio and Yang Liu. 2008. Using language models to identify language impairment in Spanish-English bilingual children. In *Proceedings of the Workshop on BioNLP 2008*, pages 116–117. ACL.

Tammie J. Spaulding, Elena Plante, and Kimberly A. Farinella. 2006. Eligibility criteria for language impairment: Is the low end of normal always appropriate? *Language, Speech, and Hearing Services in Schools*, 37(1):61–72.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Elin T. Thordardottir and Susan Ellis Weismer. 2002. Content mazes and filled pauses on narrative language samples of children with specific language impairment. *Brain and Cognition*, 48(2-3):587–592.

J. Bruce Tomblin, Nancy L. Records, Paula Buckwalter, Xuyang Zhang, Elaine Smith, and Marlea O'Brien. 1997. Prevalence of specific language impairment in kindergarten children. *Journal of Speech, Language, and Hearing Research*, 40(6):1245–1260.

Danielle Wetherell, Nicola Botting, and Gina Conti-Ramsden. 2007. Narrative in adolescent specific language impairment (SLI): a comparison with peers across two different narrative genres. *International Journal of Language and Communication Disorders*, 42:583–605(23).

Kenneth Wexler. 1994. Optional infinitives. In David Lightfoot and Norbert Hornstein, editors, *Verb Movement*. Cambridge University Press.

Ian H. Witten and Eibe Frank. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

A Discriminative Latent Variable Chinese Segmenter with Hybrid Word/Character Information

Xu Sun
Department of Computer Science
University of Tokyo
sunxu@is.s.u-tokyo.ac.jp

Yaozhong Zhang
Department of Computer Science
University of Tokyo
yaozhong.zhang@is.s.u-tokyo.ac.jp

Takuya Matsuzaki
Department of Computer Science
University of Tokyo
matuzaki@is.s.u-tokyo.ac.jp

Yoshimasa Tsuruoka
School of Computer Science
University of Manchester
yoshimasa.tsuruoka@manchester.ac.uk

Jun'ichi Tsujii
Department of Computer Science, University of Tokyo, Japan
School of Computer Science, University of Manchester, UK
National Centre for Text Mining, UK
tsujii@is.s.u-tokyo.ac.jp

Abstract

Conventional approaches to Chinese word segmentation treat the problem as a character-based tagging task. Recently, semi-Markov models have been applied to the problem, incorporating features based on complete words. In this paper, we propose an alternative, a latent variable model, which uses hybrid information based on both word sequences and character sequences. We argue that the use of latent variables can help capture long range dependencies and improve the recall on segmenting long words, e.g., named-entities. Experimental results show that this is indeed the case. With this improvement, evaluations on the data of the second SIGHAN CWS bakeoff show that our system is competitive with the best ones in the literature.

1 Introduction

For most natural language processing tasks, words are the basic units to process. Since Chinese sentences are written as continuous sequences of characters, segmenting a character sequence into a word sequence is the first step for most Chinese processing applications. In this paper, we study the problem of Chinese word segmentation (CWS), which aims to find these basic units (words[1]) for a given sentence in Chinese.

Chinese character sequences are normally ambiguous, and out-of-vocabulary (OOV) words are a major source of the ambiguity. Typical examples of OOV words include named entities (e.g., organization names, person names, and location names). Those named entities may be very long, and a difficult case occurs when a long word W ($|W| \geq 4$) consists of some words which can be separate words on their own; in such cases an automatic segmenter may split the OOV word into individual words. For example, 国际自动控制联合会计算机委员会 (Computer Committee of International Federation of Automatic Control) is one of the organization names in the Microsoft Research corpus. Its length is 13 and it contains more than 6 individual words, but it should be treated as a single word. Proper recognition of long OOV words are meaningful not only for word segmentation, but also for a variety of other purposes, e.g., full-text indexing. However, as is illustrated, recognizing long words (without sacrificing the performance on short words) is challenging.

Conventional approaches to Chinese word segmentation treat the problem as a character-based la-

[1] Following previous work, in this paper, *words* can also refer to multi-word expressions, including proper names, long named entities, idioms, etc.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 56–64,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

beling task (Xue, 2003). Labels are assigned to each character in the sentence, indicating whether the character x_i is the start ($Label_i = B$), middle or end of a multi-character word ($Label_i = C$). A popular discriminative model that have been used for this task is the conditional random fields (CRFs) (Lafferty et al., 2001), starting with the model of Peng et al. (2004). In the Second International Chinese Word Segmentation Bakeoff (the second SIGHAN CWS bakeoff) (Emerson, 2005), two of the highest scoring systems in the closed track competition were based on a CRF model (Tseng et al., 2005; Asahara et al., 2005).

While the CRF model is quite effective compared with other models designed for CWS, it may be limited by its restrictive independence assumptions on non-adjacent labels. Although the window can in principle be widened by increasing the Markov order, this may not be a practical solution, because the complexity of training and decoding a linear-chain CRF grows exponentially with the Markov order (Andrew, 2006).

To address this difficulty, a choice is to relax the Markov assumption by using the semi-Markov conditional random field model (semi-CRF) (Sarawagi and Cohen, 2004). Despite the theoretical advantage of semi-CRFs over CRFs, however, some previous studies (Andrew, 2006; Liang, 2005) exploring the use of a semi-CRF for Chinese word segmentation did not find significant gains over the CRF ones. As discussed in Andrew (2006), the reason may be that despite the greater representational power of the semi-CRF, there are some valuable features that could be more naturally expressed in a character-based labeling model. For example, on a CRF model, one might use the feature "the current character x_i is X and the current label $Label_i$ is C". This feature may be helpful in CWS for generalizing to new words. For example, it may rule out certain word boundaries if X were a character that normally occurs only as a suffix but that combines freely with some other basic forms to create new words. This type of features is slightly less natural in a semi-CRF, since in that case local features $\varphi(y_i, y_{i+1}, x)$ are defined on pairs of adjacent words. That is to say, information about which characters are not on boundaries is only implicit. Notably, except the hybrid Markov/semi-Markov system in Andrew (2006)[2], no other studies using the semi-CRF (Sarawagi and Cohen, 2004; Liang, 2005; Daumé III and Marcu, 2005) experimented with features of segmenting *non*-boundaries.

In this paper, instead of using semi-Markov models, we describe an alternative, a latent variable model, to learn long range dependencies in Chinese word segmentation. We use the discriminative probabilistic latent variable models (DPLVMs) (Morency et al., 2007; Petrov and Klein, 2008), which use latent variables to carry additional information that may not be expressed by those original labels, and therefore try to build more complicated or longer dependencies. This is especially meaningful in CWS, because the used labels are quite coarse: $Label(y) \in \{B, C\}$, where B signifies *beginning a word* and C signifies *the continuation of a word*.[3] For example, by using DPLVM, the aforementioned feature may turn to "the current character x_i is X, $Label_i = C$, and $LatentVariable_i = LV$". The current latent variable LV may strongly depend on the previous one or many latent variables, and therefore we can model the long range dependencies which may not be captured by those very coarse labels. Also, since character and word information have their different advantages in CWS, in our latent variable model, we use hybrid information based on both character and word sequences.

2 A Latent Variable Segmenter

2.1 Discriminative Probabilistic Latent Variable Model

Given data with latent structures, the task is to learn a mapping between a sequence of observations $\mathbf{x} = x_1, x_2, \ldots, x_m$ and a sequence of labels $\mathbf{y} = y_1, y_2, \ldots, y_m$. Each y_j is a class label for the j'th character of an input sequence, and is a member of a set $\mathbf{Y}$ of possible class labels. For each sequence, the model also assumes a sequence of latent variables $\mathbf{h} = h_1, h_2, \ldots, h_m$, which is unobservable in training examples.

The DPLVM is defined as follows (Morency et al.,

[2]The system was also used in Gao et al. (2007), with an improved performance in CWS.

[3]In practice, one may add a few extra labels based on linguistic intuitions (Xue, 2003).

2007):

$$P(\mathbf{y}|\mathbf{x},\mathbf{\Theta}) = \sum_{\mathbf{h}} P(\mathbf{y}|\mathbf{h},\mathbf{x},\mathbf{\Theta})P(\mathbf{h}|\mathbf{x},\mathbf{\Theta}), \quad (1)$$

where $\mathbf{\Theta}$ are the parameters of the model. DPLVMs can be seen as a natural extension of CRF models, and CRF models can be seen as a special case of DPLVMs that have only one latent variable for each label.

To make the training and inference efficient, the model is restricted to have disjoint sets of latent variables associated with each class label. Each h_j is a member in a set $\mathbf{H}_{y_j}$ of possible latent variables for the class label y_j. $\mathbf{H}$ is defined as the set of all possible latent variables, i.e., the union of all $\mathbf{H}_{y_j}$ sets. Since sequences which have any $\mathbf{h}_j \notin \mathbf{H}_{y_j}$ will by definition have $P(\mathbf{y}|\mathbf{x},\mathbf{\Theta}) = 0$, the model can be further defined[4] as:

$$P(\mathbf{y}|\mathbf{x},\mathbf{\Theta}) = \sum_{\mathbf{h}\in\mathbf{H}_{y_1}\times\ldots\times\mathbf{H}_{y_m}} P(\mathbf{h}|\mathbf{x},\mathbf{\Theta}), \quad (2)$$

where $P(\mathbf{h}|\mathbf{x},\mathbf{\Theta})$ is defined by the usual conditional random field formulation:

$$P(\mathbf{h}|\mathbf{x},\mathbf{\Theta}) = \frac{\exp \mathbf{\Theta}\cdot\mathbf{f}(\mathbf{h},\mathbf{x})}{\sum_{\forall\mathbf{h}} \exp \mathbf{\Theta}\cdot\mathbf{f}(\mathbf{h},\mathbf{x})}, \quad (3)$$

in which $\mathbf{f}(\mathbf{h},\mathbf{x})$ is a feature vector. Given a training set consisting of n labeled sequences, $(\mathbf{x}_i,\mathbf{y}_i)$, for $i = 1\ldots n$, parameter estimation is performed by optimizing the objective function,

$$L(\mathbf{\Theta}) = \sum_{i=1}^{n} \log P(\mathbf{y}_i|\mathbf{x}_i,\mathbf{\Theta}) - R(\mathbf{\Theta}). \quad (4)$$

The first term of this equation is the conditional log-likelihood of the training data. The second term is a regularizer that is used for reducing overfitting in parameter estimation.

For decoding in the test stage, given a test sequence $\mathbf{x}$, we want to find the most probable label sequence, $\mathbf{y}^*$:

$$\mathbf{y}^* = \text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x},\mathbf{\Theta}^*). \quad (5)$$

For latent conditional models like DPLVMs, the best label path $\mathbf{y}^*$ cannot directly be produced by the Viterbi algorithm because of the incorporation of hidden states. In this paper, we use a technique based on A^* search and dynamic programming described in Sun and Tsujii (2009), for producing the most probable label sequence $\mathbf{y}^*$ on DPLVM.

In detail, an A^* search algorithm[5] (Hart et al., 1968) with a Viterbi heuristic function is adopted to produce top-n latent paths, $\mathbf{h}_1, \mathbf{h}_2, \ldots \mathbf{h}_n$. In addition, a forward-backward-style algorithm is used to compute the exact probabilities of their corresponding label paths, $\mathbf{y}_1, \mathbf{y}_2, \ldots \mathbf{y}_n$. The model then tries to determine the optimal label path based on the top-n statistics, without enumerating the remaining low-probability paths, which could be exponentially enormous.

The optimal label path $\mathbf{y}^*$ is ready when the following "exact-condition" is achieved:

$$P(\mathbf{y}_1|\mathbf{x},\mathbf{\Theta}) - (1 - \sum_{\mathbf{y}_k\in\mathbf{LP}_n} P(\mathbf{y}_k|\mathbf{x},\mathbf{\Theta})) \geq 0, \quad (6)$$

where $\mathbf{y}_1$ is the most probable label sequence in current stage. It is straightforward to prove that $\mathbf{y}^* = \mathbf{y}_1$, and further search is unnecessary. This is because the remaining probability mass, $1 - \sum_{\mathbf{y}_k\in\mathbf{LP}_n} P(\mathbf{y}_k|\mathbf{x},\mathbf{\Theta})$, cannot beat the current optimal label path in this case. For more details of the inference, refer to Sun and Tsujii (2009).

2.2 Hybrid Word/Character Information

We divide our main features into two types: character-based features and word-based features. The character-based features are indicator functions that fire when the latent variable label takes some value and some predicate of the input (at a certain position) corresponding to the label is satisfied. For each latent variable label h_i (the latent variable label at position i), we use the predicate templates as follows:

- Input characters/numbers/letters locating at positions $i-2$, $i-1$, i, $i+1$ and $i+2$
- The character/number/letter bigrams locating at positions $i-2, i-1$, i and $i+1$

[4] It means that Eq. 2 is from Eq. 1 with *additional* definition.

[5] A^* search and its variants, like beam-search, are widely used in statistical machine translation. Compared to other search techniques, an interesting point of A^* search is that it can produce top-n results one-by-one in an efficient manner.

- Whether x_j and x_{j+1} are identical, for $j = (i-2) \ldots (i+1)$
- Whether x_j and x_{j+2} are identical, for $j = (i-3) \ldots (i+1)$

The latter two feature templates are designed to detect character or word reduplication, a morphological phenomenon that can influence word segmentation in Chinese.

The word-based features are indicator functions that fire when the local character sequence matches a word or a word bigram. A dictionary containing word and bigram information was collected from the training data. For each latent variable label unigram h_i, we use the set of predicate template checking for word-based features:

- The identity of the string $x_j \ldots x_i$, if it matches a word A from the word-dictionary of training data, with the constraint $i-6 < j < i$; multiple features will be generated if there are multiple strings satisfying the condition.
- The identity of the string $x_i \ldots x_k$, if it matches a word A from the word-dictionary of training data, with the constraint $i < k < i+6$; multiple features could be generated.
- The identity of the word bigram ($x_j \ldots x_{i-1}$, $x_i \ldots x_k$), if it matches a word bigram in the bigram dictionary and satisfies the aforementioned constraints on j and k; multiple features could be generated.
- The identity of the word bigram ($x_j \ldots x_i$, $x_{i+1} \ldots x_k$), if it matches a word bigram in the bigram dictionary and satisfies the aforementioned constraints on j and k; multiple features could be generated.

All feature templates were instantiated with values that occur in positive training examples. We found that using low-frequency features that occur only a few times in the training set improves performance on the development set. We hence do not do any thresholding of the DPLVM features: we simply use all those generated features.

The aforementioned word based features can incorporate word information naturally. In addition, following Wang et al. (2006), we found using a very simple heuristic can further improve the segmentation quality slightly. More specifically, two operations, *merge* and *split*, are performed on the DPLVM/CRF outputs: if a bigram $A \_ B$ was not observed in the training data, but the merged one AB was, then $A \_ B$ will be simply merged into AB; on the other hand, if AB was not observed but $A \_ B$ appeared, then it will be split into $A \_ B$. We found this simple heuristic on word information slightly improved the performance (e.g., for the PKU corpus, +0.2% on the F-score).

3 Experiments

We used the data provided by the second International Chinese Word Segmentation Bakeoff to test our approaches described in the previous sections. The data contains three corpora from different sources: Microsoft Research Asia (MSR), City University of Hong Kong (CU), and Peking University (PKU).

Since the purpose of this work is to evaluate the proposed latent variable model, we did not use extra resources such as common surnames, lexicons, parts-of-speech, and semantics. For the generation of word-based features, we extracted a word list from the training data as the vocabulary.

Four metrics were used to evaluate segmentation results: recall (R, the percentage of gold standard output words that are correctly segmented by the decoder), precision (P, the percentage of words in the decoder output that are segmented correctly), balanced F-score (F) defined by $2PR/(P+R)$, recall of OOV words (R-oov). For more detailed information on the corpora and these metrics, refer to Emerson (2005).

3.1 Training the DPLVM Segmenter

We implemented DPLVMs in C++ and optimized the system to cope with large scale problems, in which the feature dimension is beyond millions. We employ the feature templates defined in Section 2.2, taking into account those 3,069,861 features for the MSR data, 2,634,384 features for the CU data, and 1,989,561 features for the PKU data.

As for numerical optimization, we performed gradient decent with the Limited-Memory BFGS

(L-BFGS)[6] optimization technique (Nocedal and Wright, 1999). L-BFGS is a second-order Quasi-Newton method that numerically estimates the curvature from previous gradients and updates. With no requirement on specialized Hessian approximation, L-BFGS can handle large-scale problems in an efficient manner.

Since the objective function of the DPLVM model is non-convex, we randomly initialized parameters for the training.[7] To reduce overfitting, we employed an L_2 Gaussian weight prior[8] (Chen and Rosenfeld, 1999). During training, we varied the L_2-regularization term (with values 10^k, k from -3 to 3), and finally set the value to 1. We use 4 hidden variables per label for this task, compromising between accuracy and efficiency.

3.2 Comparison on Convergence Speed

First, we show a comparison of the convergence speed between the objective function of DPLVMs and CRFs. We apply the L-BFGS optimization algorithm to optimize the objective function of DPLVM and CRF models, making a comparison between them. We find that the number of iterations required for the convergence of DPLVMs are fewer than for CRFs. Figure 1 illustrates the convergence-speed comparison on the MSR data. The DPLVM model arrives at the plateau of convergence in around 300 iterations, with the penalized loss of $95K$ when $\#passes = 300$; while CRFs require 900 iterations, with the penalized loss of $98K$ when $\#passes = 900$.

However, we should note that the time cost of the DPLVM model in each iteration is around four times higher than the CRF model, because of the incorporation of hidden variables. In order to speed up the training speed of the DPLVM model in the future, one solution is to use the stochastic learning technique[9]. Another solution is to use a distributed version of L-BFGS to parallelize the batch training.

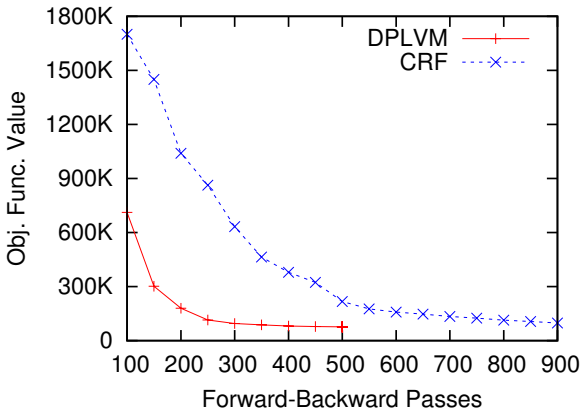


Figure 1: The value of the penalized loss based on the number of iterations: DPLVMs vs. CRFs on the MSR data.

	Style	#W.T.	#Word	#C.T.	#Char
MSR	S.C.	88K	2,368K	5K	4,050K
CU	T.C.	69K	1,455K	5K	2,403K
PKU	S.C.	55K	1,109K	5K	1,826K

Table 1: Details of the corpora. *W.T.* represents *word types*; *C.T.* represents *character types*; *S.C.* represents *simplified Chinese*; *T.C.* represents *traditional Chinese*.

4 Results and Discussion

Since the CRF model is one of the most successful models in Chinese word segmentation, we compared DPLVMs with CRFs. We tried to make experimental results comparable between DPLVMs and CRF models, and have therefore employed the same feature set, optimizer and fine-tuning strategy between the two. We also compared DPLVMs with semi-CRFs and other successful systems reported in previous work.

4.1 Evaluation Results

Three training and test corpora were used in the test, including the MSR Corpus, the CU Corpus, and the

[6]For numerical optimization on latent variable models, we also experimented the conjugate-gradient (CG) optimization algorithm and stochastic gradient decent algorithm (SGD). We found the L-BFGS with L_2 Gaussian regularization performs slightly better than the CG and the SGD. Therefore, we adopt the L-BFGS optimizer in this study.

[7]For a non-convex objective function, different parameter initializations normally bring different optimization results. Therefore, to approach closer to the global optimal point, it is recommended to perform multiple experiments on DPLVMs with random initialization and then select a good start point.

[8]We also tested the L-BFGS with L_1 regularization, and we found the L-BFGS with L_2 regularization performs better in this task.

[9]We have tried stochastic gradient decent, as described previously. It is possible to try other stochastic learning methods, e.g., stochastic meta decent (Vishwanathan et al., 2006).

MSR data	P	R	F	R-oov
DPLVM (*)	97.3	97.3	**97.3**	**72.2**
CRF (*)	97.1	96.8	97.0	72.0
semi-CRF (A06)	N/A	N/A	96.8	N/A
semi-CRF (G07)	N/A	N/A	97.2	N/A
CRF (Z06-a)	96.5	96.3	96.4	71.4
Z06-b	97.2	96.9	97.1	71.2
ZC07	N/A	N/A	97.2	N/A
Best05 (T05)	96.2	96.6	96.4	71.7
CU data	P	R	F	R-oov
DPLVM (*)	94.7	94.4	94.6	68.8
CRF (*)	94.3	93.9	94.1	65.8
CRF (Z06-a)	95.0	94.2	94.6	73.6
Z06-b	95.2	94.9	**95.1**	**74.1**
ZC07	N/A	N/A	**95.1**	N/A
Best05 (T05)	94.1	94.6	94.3	69.8
PKU data	P	R	F	R-oov
DPLVM (*)	95.6	94.8	**95.2**	**77.8**
CRF (*)	95.2	94.2	94.7	76.8
CRF (Z06-a)	94.3	94.6	94.5	75.4
Z06-b	94.7	95.5	95.1	74.8
ZC07	N/A	N/A	94.5	N/A
Best05 (C05)	95.3	94.6	95.0	63.6

Table 2: Results from DPLVMs, CRFs, semi-CRFs, and other systems.

PKU Corpus (see Table 1 for details). The results are shown in Table 2. The results are grouped into three sub-tables according to different corpora. Each row represents a CWS model. For each group, the rows marked by $*$ represent our models with hybrid word/character information. *Best05* represents the best system of the Second International Chinese Word Segmentation Bakeoff on the corresponding data; *A06* represents the semi-CRF model in Andrew (2006)[10], which was also used in Gao et al. (2007) (denoted as *G07*) with an improved performance; *Z06-a* and *Z06-b* represents the pure subword CRF model and the confidence-based combination of CRF and rule-based models, respectively (Zhang et al., 2006); *ZC07* represents the word-based perceptron model in Zhang and Clark (2007); *T05* represents the CRF model in Tseng et al. (2005); *C05* represents the system in Chen et al. (2005). The best F-score and recall of OOV words of each group is shown in bold.

As is shown in the table, we achieved the best F-score in two out of the three corpora. We also achieved the best recall rate of OOV words on those two corpora. Both of the MSR and PKU Corpus use simplified Chinese, while the CU Corpus uses the traditional Chinese.

On the MSR Corpus, the DPLVM model reduced more than 10% error rate over the CRF model using exactly the same feature set. We also compared our DPLVM model with the semi-CRF models in Andrew (2006) and Gao et al. (2007), and demonstrate that the DPLVM model achieved slightly better performance than the semi-CRF models. Andrew (2006) and Gao et al. (2007) only reported the results on the MSR Corpus.

In summary, tests for the Second International Chinese Word Segmentation Bakeoff showed competitive results for our method compared with the best results in the literature. Our discriminative latent variable models achieved the best F-scores on the MSR Corpus (97.3%) and PKU Corpus (95.2%); the latent variable models also achieved the best recalls of OOV words over those two corpora. We will analyze the results by varying the word-length in the following subsection.

4.2 Effect on Long Words

One motivation of using a latent variable model for CWS is to use latent variables to more adequately learn long range dependencies, as we argued in Section 1. In the test data of the MSR Corpus, 19% of the words are longer than 3 characters; there are also 8% in the CU Corpus and 11% in the PKU Corpus, respectively. In the MSR Corpus, there are some extremely long words ($Length > 10$), while the CU and PKU corpus do not contain such extreme cases.

Figure 2 shows the recall rate on different groups of words categorized by their lengths (the number of characters). As we expected, the DPLVM model performs much better on long words ($Length \geq 4$) than the CRF model, which used exactly the same feature set. Compared with the CRF model, the DPLVM model exhibited almost the same level of performance on short words. Both models have the best performance on segmenting the words with the length of two. The performance of the CRF

[10]It is a hybrid Markov/semi-Markov CRF model which outperforms conventional semi-CRF models (Andrew, 2006). However, in general, as discussed in Andrew (2006), it is essentially still a semi-CRF model.

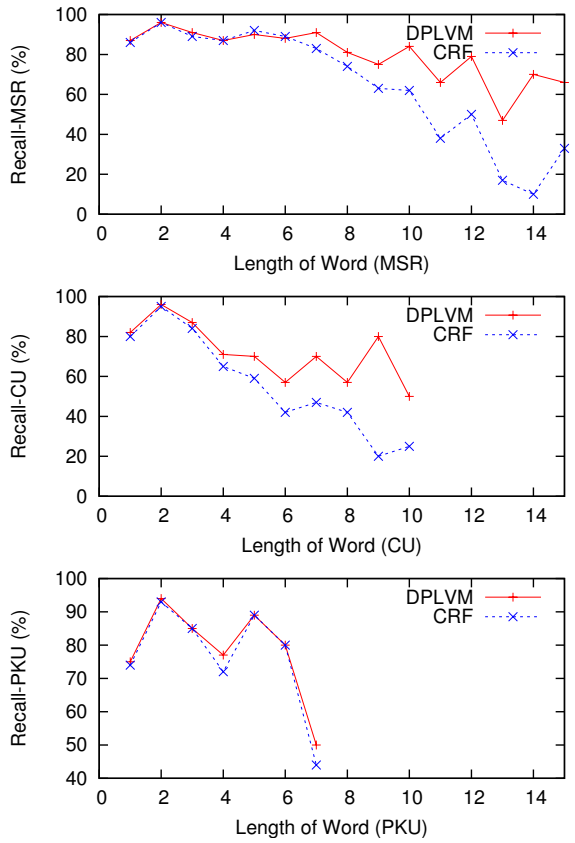


Figure 2: The recall rate on words grouped by the length.

model deteriorates rapidly as the word length increases, which demonstrated the difficulty on modeling long range dependencies in CWS. Compared with the CRF model, the DPLVM model performed quite well in dealing with long words, without sacrificing the performance on short words. All in all, we conclude that the improvement of using the DPLVM model came from the improvement on modeling long range dependencies in CWS.

4.3 Error Analysis

Table 3 lists the major errors collected from the latent variable segmenter. We examined the collected errors and found that many of them can be grouped into four types: over-generalization (the top row), errors on named entities (the following three rows), errors on idioms (the following three rows) and errors from inconsistency (the two rows at the bottom).

Our system performed reasonably well on very complex OOV words, such as 中国农业银行石家庄分行第二营业部 (Agricultural Bank of China, Shijiazhuang-city Branch, the second sales department) and 国家科委中国科技信息研究所 (Science and Technology Commission of China, National Institution on Scientific Information Analysis). However, it sometimes over-generalized to long words. For example, as shown in the top row, 国家环保局 (National Department of Environmental Protection) and 中宣部 (The Central Propaganda Department) are two organization names, but they are incorrectly merged into a single word.

Gold Segmentation	Segmenter Output
国家环保局//中宣部	国家环保局中宣部
Co-allocated org. names	
陈耀 (Chen Yao)	陈//耀
陈飞 (Chen Fei)	陈//飞
瓦西里斯 (Vasillis)	瓦//西里斯
通宵达旦	通宵//达旦
好高骛远	好//高骛远
一蹶不振	一//蹶//不振
Idioms	
宣传//家 (propagandist)	宣传家
沙漠化 (desertification)	沙漠//化

Table 3: Error analysis on the latent variable segmenter. The errors are grouped into four types: over-generalization, errors on named entities, errors on idioms and errors from data-inconsistency.

As for the following three rows, 陈耀 (Chen Yao) and 陈飞 (Chen Fei) are person names. They are wrongly segmented because we lack the features to capture the information of person names (such useful knowledge, e.g., common surname list, are currently not used in our system). In the future, such errors may be solved by integrating open resources into our system. 瓦西里斯 (Vasillis) is a transliterated foreign location name and is also wrongly segmented.

For the corpora that considered 4 character idioms as a word, our system successfully combined most of new idioms together. This differs greatly from the results of CRFs. However, there are still a number of new idioms that failed to be correctly segmented, as listed from the fifth row to the seventh row.

Finally, some errors are due to inconsistencies in the gold segmentation. For example, 宣传//家 (propagandist) is two words, but a word with similar

structure, 理论家 (theorist), is one word. 沙漠化 (desertification) is one word, but its synonym, 荒漠//化 (desertification), is two words in the gold segmentation.

5 Conclusion and Future Work

We presented a latent variable model for Chinese word segmentation, which used hybrid information based on both word and character sequences. We discussed that word and character information have different advantages, and could be complementary to each other. Our model is an alternative to the existing word based models and character based models.

We argued that using latent variables can better capture long range dependencies. We performed experiments and demonstrated that our model can indeed improve the segmentation accuracy on long words. With this improvement, tests on the data of the Second International Chinese Word Segmentation Bakeoff show that our system is competitive with the best in the literature.

Since the latent variable model allows a wide range of features, so the future work will consider how to integrate open resources into our system. The latent variable model handles latent-dependencies naturally, and can be easily extended to other labeling tasks.

Acknowledgments

We thank Kun Yu, Galen Andrew and Xiaojun Lin for the enlightening discussions. We also thank the anonymous reviewers who gave very helpful comments. This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan).

References

Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation. *Proceedings of EMNLP'06*, pages 465–472.

Masayuki Asahara, Kenta Fukuoka, Ai Azuma, Chooi-Ling Goh, Yotaro Watanabe, Yuji Matsumoto, and Takahashi Tsuzuki. 2005. Combination of machine learning methods for optimum chinese word segmentation. *Proceedings of the fourth SIGHAN workshop*, pages 134–137.

Stanley F. Chen and Ronald Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. *Technical Report CMU-CS-99-108, CMU*.

Aitao Chen, Yiping Zhou, Anne Zhang, and Gordon Sun. 2005. Unigram language model for chinese word segmentation. *Proceedings of the fourth SIGHAN workshop*.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. *Proceedings of ICML'05*.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. *Proceedings of the fourth SIGHAN workshop*, pages 123–133.

Jianfeng Gao, Galen Andrew, Mark Johnson, and Kristina Toutanova. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*, pages 824–831.

P.E. Hart, N.J. Nilsson, and B. Raphael. 1968. A formal basis for the heuristic determination of minimum cost path. *IEEE Trans. On System Science and Cybernetics*, SSC-4(2):100–107.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of ICML'01*, pages 282–289.

Percy Liang. 2005. Semi-supervised learning for natural language. *Master's thesis, Massachusetts Institute of Technology*.

Louis-Philippe Morency, Ariadna Quattoni, and Trevor Darrell. 2007. Latent-dynamic discriminative models for continuous gesture recognition. *Proceedings of CVPR'07*, pages 1–8.

Jorge Nocedal and Stephen J. Wright. 1999. Numerical optimization. *Springer*.

F. Peng and A. McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. *Proceedings of COLING'04*.

Slav Petrov and Dan Klein. 2008. Discriminative log-linear grammars with latent variables. *Proceedings of NIPS'08*.

Sunita Sarawagi and William Cohen. 2004. Semi-markov conditional random fields for information extraction. *Proceedings of ICML'04*.

Xu Sun and Jun'ichi Tsujii. 2009. Sequential labeling with latent variables: An exact inference algorithm and its efficient approximation. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL'09)*.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff

2005. *Proceedings of the fourth SIGHAN workshop*, pages 168–171.

S.V.N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. 2006. Accelerated training of conditional random fields with stochastic meta-descent. *Proceedings of ICML'06*, pages 969–976.

Xinhao Wang, Xiaojun Lin, Dianhai Yu, Hao Tian, and Xihong Wu. 2006. Chinese word segmentation with maximum entropy and n-gram language model. In *Proceedings of the fifth SIGHAN workshop*, pages 138–141, July.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *International Journal of Computational Linguistics and Chinese Language Processing*, 8(1).

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. *Proceedings of ACL'07*.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging by conditional random fields for chinese word segmentation. *Proceedings of HLT/NAACL'06 companion volume short papers*.

Improved Reconstruction of Protolanguage Word Forms

Alexandre Bouchard-Côté* **Thomas L. Griffiths†** **Dan Klein***
*Computer Science Division †Department of Psychology
University of California at Berkeley
Berkeley, CA 94720

Abstract

We present an unsupervised approach to reconstructing ancient word forms. The present work addresses three limitations of previous work. First, previous work focused on *faithfulness* features, which model changes between successive languages. We add *markedness* features, which model well-formedness within each language. Second, we introduce universal features, which support generalizations across languages. Finally, we increase the number of languages to which these methods can be applied by an order of magnitude by using improved inference methods. Experiments on the reconstruction of Proto-Oceanic, Proto-Malayo-Javanic, and Classical Latin show substantial reductions in error rate, giving the best results to date.

1 Introduction

A central problem in diachronic linguistics is the reconstruction of ancient languages from their modern descendants (Campbell, 1998). Here, we consider the problem of reconstructing phonological forms, given a known linguistic phylogeny and known cognate groups. For example, Figure 1 (a) shows a collection of word forms in several Oceanic languages, all meaning *to cry*. The ancestral form in this case has been presumed to be /taŋis/ in Blust (1993). We are interested in models which take as input many such word tuples, each representing a cognate group, along with a language tree, and induce word forms for hidden ancestral languages.

The traditional approach to this problem has been the *comparative method*, in which reconstructions are done manually using assumptions about the relative probability of different kinds of sound change (Hock, 1986). There has been work attempting to automate part (Durham and Rogers, 1969; Eastlack, 1977; Lowe and Mazaudon, 1994; Covington, 1998; Kondrak, 2002) or all of the process (Oakes, 2000; Bouchard-Côté et al., 2008). However, previous automated methods have been unable to leverage three important ideas a linguist would employ. We address these omissions here, resulting in a more powerful method for automatically reconstructing ancient protolanguages.

First, linguists triangulate reconstructions from many languages, while past work has been limited to small numbers of languages. For example, Oakes (2000) used four languages to reconstruct Proto-Malayo-Javanic (PMJ) and Bouchard-Côté et al. (2008) used two languages to reconstruct Classical Latin (La). We revisit these small datasets and show that our method significantly outperforms these previous systems. However, we also show that our method can be applied to a much larger data set (Greenhill et al., 2008), reconstructing Proto-Oceanic (POc) from 64 modern languages. In addition, performance *improves* with more languages, which was not the case for previous methods.

Second, linguists exploit knowledge of phonological universals. For example, small changes in vowel height or consonant place are more likely than large changes, and much more likely than change to arbitrarily different phonemes. In a statistical system, one could imagine either manually encoding or automatically inferring such preferences. We show that both strategies are effective.

Finally, linguists consider not only how languages change, but also how they are internally consistent. Past models described how sounds do (or, more often, do not) change between nodes in the tree. To borrow broad terminology from the Optimality Theory literature (Prince and Smolensky, 1993), such models incorporated *faithfulness* features, capturing the ways in which successive forms remained similar to one another. However, each language has certain regular phonotactic patterns which con-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 65–73,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

strain these changes. We encode such patterns using *markedness* features, characterizing the internal phonotactic structure of each language. Faithfulness and markedness play roles analogous to the channel and language models of a noisy-channel system. We show that markedness features improve reconstruction, and can be used efficiently.

2 Related work

Our focus in this section is on describing the properties of the two previous systems for reconstructing ancient word forms to which we compare our method. Citations for other related work, such as similar approaches to using faithfulness and markedness features, appear in the body of the paper.

In Oakes (2000), the word forms in a given protolanguage are reconstructed using a Viterbi multi-alignment between a small number of its descendant languages. The alignment is computed using hand-set parameters. Deterministic rules characterizing changes between pairs of observed languages are extracted from the alignment when their frequency is higher than a threshold, and a proto-phoneme inventory is built using linguistically motivated rules and parsimony. A reconstruction of each observed word is first proposed independently for each language. If at least two reconstructions agree, a majority vote is taken, otherwise no reconstruction is proposed. This approach has several limitations. First, it is not tractable for larger trees, since the time complexity of their multi-alignment algorithm grows exponentially in the number of languages. Second, deterministic rules, while elegant in theory, are not robust to noise: even in experiments with only four daughter languages, a large fraction of the words could not be reconstructed.

In Bouchard-Côté et al. (2008), a stochastic model of sound change is used and reconstructions are inferred by performing probabilistic inference over an evolutionary tree expressing the relationships between languages. The model does not support generalizations across languages, and has no way to capture phonotactic regularities within languages. As a consequence, the resulting method does not scale to large phylogenies. The work we present here addresses both of these issues, with a richer model and faster inference allowing improved reconstruction and increased scale.

3 Model

We start this section by introducing some notation. Let τ be a tree of languages, such as the examples in Figure 3 (c-e). In such a tree, the modern languages, whose word forms will be observed, are the leaves of τ. All internal nodes, particularly the root, are languages whose word forms are not observed. Let L denote all languages, modern and otherwise. All word forms are assumed to be strings Σ^* in the International Phonological Alphabet (IPA).[1]

We assume that word forms evolve along the branches of the tree τ. However, it is not the case that each cognate set exists in each modern language. Formally, we assume there to be a known list of C cognate sets. For each $c \in \{1, \ldots, C\}$ let $L(c)$ denote the subset of modern languages that have a word form in the c-th cognate set. For each set $c \in \{1, \ldots, C\}$ and each language $\ell \in L(c)$, we denote the modern word form by $w_{c\ell}$. For cognate set c, only the minimal subtree $\tau(c)$ containing $L(c)$ and the root is relevant to the reconstruction inference problem for that set.

From a high-level perspective, the generative process is quite simple. Let c be the index of the current cognate set, with topology $\tau(c)$. First, a word is generated for the root of $\tau(c)$ using an (initially unknown) root language model (distribution over strings). The other nodes of the tree are drawn incrementally as follows: for each edge $\ell \rightarrow \ell'$ in $\tau(c)$ use a branch-specific distribution over changes in strings to generate the word at node ℓ'.

In the remainder of this section, we clarify the exact form of the conditional distributions over string changes, the distribution over strings at the root, and the parameterization of this process.

3.1 Markedness and Faithfulness

In Optimality Theory (OT) (Prince and Smolensky, 1993), two types of constraints influence the selection of a realized output given an input form: *faithfulness* and *markedness* constraints. Faithfulness en-

[1] The choice of a phonemic representation is motivated by the fact that most of the data available comes in this form. Diacritics are available in a smaller number of languages and may vary across dialects, so we discarded them in this work.

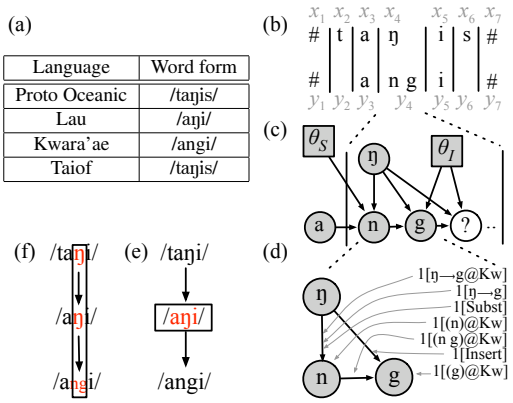


Figure 1: (a) A cognate set from the Austronesian dataset. All word forms mean *to cry*. (b-d) The mutation model used in this paper. (b) The mutation of POc /taŋis/ to Kw. /angi/. (c) Graphical model depicting the dependencies among variables in one step of the mutation Markov chain. (d) Active features for one step in this process. (e-f) Comparison of two inference procedures on trees: Single sequence resampling (e) draws one sequence at a time, conditioned on its parent and children, while ancestry resampling (f) draws an aligned slice from all words simultaneously. In large trees, the latter is more efficient than the former.

courages similarity between the input and output while markedness favors well-formed output.

Viewed from this perspective, previous computational approaches to reconstruction are based almost exclusively on faithfulness, expressed through a mutation model. Only the words in the language at the root of the tree, if any, are explicitly encouraged to be well-formed. In contrast, we incorporate constraints on markedness for each language with both general and branch-specific constraints on faithfulness. This is done using a *lexicalized stochastic string transducer* (Varadarajan et al., 2008).

We now make precise the conditional distributions over pairs of evolving strings, referring to Figure 1 (b-d). Consider a language ℓ' evolving to ℓ for cognate set c. Assume we have a word form $x = w_{c\ell'}$. The generative process for producing $y = w_{c\ell}$ works as follows. First, we consider x to be composed of characters $x_1 x_2 \ldots x_n$, with the first and last being a special boundary symbol $x_1 = \# \in \Sigma$ which is never deleted, mutated, or created. The process generates $y = y_1 y_2 \ldots y_n$ in n chunks $y_i \in \Sigma^*, i \in \{1, \ldots, n\}$, one for each x_i. The y_i's may be a single character, multiple characters, or even empty. In the example shown, all three of these cases occur.

To generate y_i, we define a *mutation Markov chain* that incrementally adds zero or more characters to an initially empty y_i. First, we decide whether the current phoneme in the top word $t = x_i$ will be deleted, in which case $y_i = \epsilon$ as in the example of /s/ being deleted. If t is not deleted, we chose a single substitution character in the bottom word. This is the case both when /a/ is unchanged and when /ŋ/ substitutes to /n/. We write $\mathscr{S} = \Sigma \cup \{\zeta\}$ for this set of outcomes, where ζ is the special outcome indicating deletion. Importantly, the probabilities of this multinomial can depend on both the previous character generated so far (i.e. the rightmost character p of y_{i-1}) and the current character in the previous generation string (t). As we will see shortly, this allows modelling markedness and faithfulness at every branch, jointly. This multinomial decision acts as the initial distribution of the mutation Markov chain.

We consider insertions only if a deletion was not selected in the first step. Here, we draw from a multinomial over $\mathscr{S}$, where this time the special outcome ζ corresponds to stopping insertions, and the other elements of $\mathscr{S}$ correspond to symbols that are appended to y_i. In this case, the conditioning environment is $t = x_i$ and the current rightmost symbol p in y_i. Insertions continue until ζ is selected. In the example, we follow the substitution of /ŋ/ to /n/ with an insertion of /g/, followed by a decision to stop that y_i. We will use $\theta_{S,t,p,\ell}$ and $\theta_{I,t,p,\ell}$ to denote the probabilities over the substitution and insertion decisions in the current branch $\ell' \rightarrow \ell$.

A similar process generates the word at the root ℓ of a tree, treating this word as a single string y_1 generated from a dummy ancestor $t = x_1$. In this case, only the insertion probabilities matter, and we separately parameterize these probabilities with $\theta_{R,t,p,\ell}$. There is no actual dependence on t at the root, but this formulation allows us to unify the parameterization, with each $\theta_{\omega,t,p,\ell} \in \mathbb{R}^{|\Sigma|+1}$ where $\omega \in \{R, S, I\}$.

3.2 Parameterization

Instead of directly estimating the transition probabilities of the mutation Markov chain (as the parameters of a collection of multinomial distributions) we

express them as the output of a log-linear model. We used the following feature templates:

OPERATION identifies whether an operation in the mutation Markov chain is an insertion, a deletion, a substitution, a self-substitution (i.e. of the form $x \rightarrow y, x = y$), or the end of an insertion event. Examples in Figure 1 (d): $\mathbf{1}[\text{Subst}]$ and $\mathbf{1}[\text{Insert}]$.

MARKEDNESS consists of language-specific n-gram indicator functions for all symbols in Σ. Only unigram and bigram features are used for computational reasons, but we show in Section 5 that this already captures important constraints. Examples in Figure 1 (d): the bigram indicator $\mathbf{1}[\text{(n g)@Kw}]$ (Kw stands for Kwara'ae, a language of the Solomon Islands), the unigram indicators $\mathbf{1}[\text{(n)@Kw}]$ and $\mathbf{1}[\text{(g)@Kw}]$.

FAITHFULNESS consists of indicators for mutation events of the form $\mathbf{1}[x \rightarrow y]$, where $x \in \Sigma$, $y \in \mathscr{S}$. Examples: $\mathbf{1}[\text{ŋ} \rightarrow \text{n}]$, $\mathbf{1}[\text{ŋ} \rightarrow \text{n@Kw}]$.

Feature templates similar to these can be found for instance in Dreyer et al. (2008) and Chen (2003), in the context of string-to-string transduction. Note also the connection with stochastic OT (Goldwater and Johnson, 2003; Wilson, 2006), where a log-linear model mediates markedness and faithfulness of the production of an output form from an underlying input form.

3.3 Parameter sharing

Data sparsity is a significant challenge in protolanguage reconstruction. While the experiments we present here use an order of magnitude more languages than previous computational approaches, the increase in observed data also brings with it additional unknowns in the form of intermediate protolanguages. Since there is one set of parameters for each language, adding more data is not sufficient for increasing the quality of the reconstruction: we show in Section 5.2 that adding extra languages can actually hurt reconstruction using previous methods. It is therefore important to share parameters across different branches in the tree in order to benefit from having observations from more languages.

As an example of useful parameter sharing, consider the faithfulness features $\mathbf{1}[\text{/p/} \rightarrow \text{/b/}]$ and $\mathbf{1}[\text{/p/} \rightarrow \text{/r/}]$, which are indicator functions for the appearance of two substitutions for /p/. We would like the model to learn that the former event (a simple voicing change) should be preferred over the latter. In Bouchard-Côté et al. (2008), this has to be learned for each branch in the tree. The difficulty is that not all branches will have enough information to learn this preference, meaning that we need to define the model in such a way that it can generalize across languages.

We used the following technique to address this problem: we augment the sufficient statistics of Bouchard-Côté et al. (2008) to include the current language (or language at the bottom of the current branch) and use a single, global weight vector instead of a set of branch-specific weights. Generalization across branches is then achieved by using features that *ignore* ℓ, while branch-specific features depend on ℓ.

For instance, in Figure 1 (d), $\mathbf{1}[\text{ŋ} \rightarrow \text{n}]$ is an example of a universal (global) feature shared across all branches while $\mathbf{1}[\text{ŋ} \rightarrow \text{n@Kw}]$ is branch-specific. Similarly, all of the features in OPERATION, MARKEDNESS and FAITHFULNESS have universal and branch-specific versions.

3.4 Objective function

Concretely, the transition probabilities of the mutation and root generation are given by:

$$\theta_{\omega,t,p,\ell}(\xi) = \frac{\exp\{\langle \lambda, f(\omega,t,p,\ell,\xi)\rangle\}}{Z(\omega,t,p,\ell,\lambda)} \times \mu(\omega,t,\xi),$$

where $\xi \in \mathscr{S}$, $f : \{S,I,R\} \times \Sigma \times \Sigma \times L \times \mathscr{S} \rightarrow \mathbb{R}^k$ is the sufficient statistics or feature function, $\langle \cdot, \cdot \rangle$ denotes inner product and $\lambda \in \mathbb{R}^k$ is a weight vector. Here, k is the dimensionality of the feature space of the log-linear model. In the terminology of exponential families, Z and μ are the normalization function and reference measure respectively:

$$Z(\omega,t,p,\ell,\lambda) = \sum_{\xi' \in \mathscr{S}} \exp\{\langle \lambda, f(\omega,t,p,\ell,\xi')\rangle\}$$

$$\mu(\omega,t,\xi) = \begin{cases} 0 \text{ if } \omega = S, t = \#, \xi \neq \# \\ 0 \text{ if } \omega = R, \xi = \zeta \\ 0 \text{ if } \omega \neq R, \xi = \# \\ 1 \text{ o.w.} \end{cases}$$

Here, μ is used to handle boundary conditions.

We will also need the following notation: let $\mathbb{P}_\lambda(\cdot), \mathbb{P}_\lambda(\cdot|\cdot)$ denote the root and branch probability models described in Section 3.1 (with transition probabilities given by the above log-linear model), $I(c)$, the set of internal (non-leaf) nodes in $\tau(c)$, $\mathrm{pa}(\ell)$, the parent of language ℓ, $\mathrm{r}(c)$, the root of $\tau(c)$

and $W(c) = (\Sigma^*)^{|I(c)|}$. We can summarize our objective function as follows:

$$\sum_{c=1}^{C} \log \sum_{\vec{w} \in W(c)} \mathbb{P}_\lambda(w_{c,\mathrm{r}(c)}) \prod_{\ell \in I(c)} \mathbb{P}_\lambda(w_{c,\ell} | w_{c,\mathrm{pa}(\ell)}) - \frac{||\lambda||_2^2}{2\sigma^2}$$

The second term is a standard L^2 regularization penalty (we used $\sigma^2 = 1$).

4 Learning algorithm

Learning is done using a Monte Carlo variant of the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). The M step is convex and computed using L-BFGS (Liu et al., 1989); but the E step is intractable (Lunter et al., 2003), so we used a Markov chain Monte Carlo (MCMC) approximation (Tierney, 1994). At E step $t = 1, 2, \ldots$, we simulated the chain for $O(t)$ iterations; this regime is necessary for convergence (Jank, 2005).

In the E step, the inference problem is to compute an expectation under the posterior over strings in a protolanguage given observed word forms at the leaves of the tree. The typical approach in biology or historical linguistics (Holmes and Bruno, 2001; Bouchard-Côté et al., 2008) is to use Gibbs sampling, where the entire string at a single node in the tree is sampled, conditioned on its parent and children. This sampling domain is shown in Figure 1 (e), where the middle word is completely resampled but adjacent words are fixed. We will call this method Single Sequence Resampling (SSR). While conceptually simple, this approach suffers from problems in large trees (Holmes and Bruno, 2001). Consequently, we use a different MCMC procedure, called Ancestry Resampling (AR) that alleviates the mixing problems (Figure 1 (f)). This method was originally introduced for biological applications (Bouchard-Côté et al., 2009), but commonalities between the biological and linguistic cases make it possible to use it in our model.

Concretely, the problem with SSR arises when the tree under consideration is large or unbalanced. In this case, it can take a long time for information from the observed languages to propagate to the root of the tree. Indeed, samples at the root will initially be *independent* of the observations. AR addresses this problem by resampling one thin vertical slice of all sequences at a time, called an ancestry. For the precise definition, see Bouchard-Côté et al. (2009). Slices condition on observed data, avoiding the problems mentioned above, and can propagate information rapidly across the tree.

5 Experiments

We performed a comprehensive set of experiments to test the new method for reconstruction outlined above. In Section 5.1, we analyze in isolation the effects of varying the set of features, the number of observed languages, the topology, and the number of iterations of EM. In Section 5.2 we compare performance to an oracle and to three other systems.

Evaluation of all methods was done by computing the Levenshtein distance (Levenshtein, 1966) between the reconstruction produced by each method and the reconstruction produced by linguists. We averaged this distance across reconstructed words to report a single number for each method. We show in Table 2 the average word length in each corpus; note that the Latin average is much larger, giving an explanation to the higher errors in the Romance dataset. The statistical significance of all performance differences are assessed using a paired t-test with significance level of 0.05.

5.1 Evaluating system performance

We used the Austronesian Basic Vocabulary Database (Greenhill et al., 2008) as the basis for a series of experiments used to evaluate the performance of our system and the factors relevant to its success. The database includes partial cognacy judgments and IPA transcriptions, as well as a few reconstructed protolanguages. A reconstruction of Proto-Oceanic (POc) originally developed by Blust (1993) using the comparative method was the basis for evaluation.

We used the cognate information provided in the database, automatically constructing a global tree[2] and set of subtrees from the cognate set indicator matrix $M(\ell, c) = \mathbf{1}[\ell \in L(c)], c \in \{1, \ldots, C\}, \ell \in L$. For constructing the global tree, we used the implementation of neighbor joining in the Phylip package (Felsenstein, 1989). We used a distance based on cognates overlap, $d_c(\ell_1, \ell_2) = \sum_{c=1}^{C} M(\ell_1, c) M(\ell_2, c)$. We bootstrapped 1000

[2]The dataset included a tree, but it was out of date as of November 2008 (Greenhill et al., 2008).

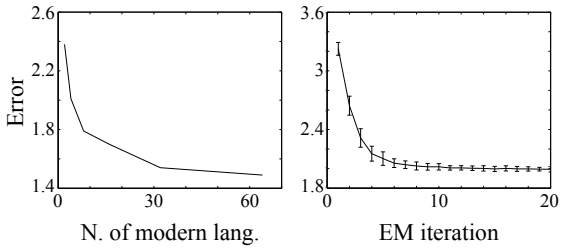


Figure 2: Left: Mean distance to the target reconstruction of POc as a function of the number of modern languages used by the inference procedure. Right: Mean distance and confidence intervals as a function of the EM iteration, averaged over 20 random seeds and ran on 4 languages.

Condition	Edit dist.
Unsupervised full system	1.87
-FAITHFULNESS	2.02
-MARKEDNESS	2.18
-Sharing	1.99
-Topology	2.06
Semi-supervised system	1.75

Table 1: Effects of ablation of various aspects of our unsupervised system on mean edit distance to POc. -Sharing corresponds to the restriction to the subset of the features in OPERATION, FAITHFULNESS and MARKEDNESS that are branch-specific, -Topology corresponds to using a flat topology where the only edges in the tree connect modern languages to POc. The semi-supervised system is described in the text. All differences (compared to the unsupervised full system) are statistically significant.

samples and formed an accurate (90%) consensus tree. The tree obtained is not binary, but the AR inference algorithm scales linearly in the branching factor of the tree (in contrast, SSR scales exponentially (Lunter et al., 2003)).

The first claim we verified experimentally is that having more observed languages aids reconstruction of protolanguages. To test this hypothesis we added observed modern languages in increasing order of distance d_c to the target reconstruction of POc so that the languages that are most useful for POc reconstruction are added first. This prevents the effects of adding a close language after several distant ones being confused with an improvement produced by increasing the number of languages.

The results are reported in Figure 2 (a). They confirm that large-scale inference is desirable for automatic protolanguage reconstruction: reconstruction improved statistically significantly with each increase except from 32 to 64 languages, where the average edit distance improvement was 0.05.

We then conducted a number of experiments intended to assess the robustness of the system, and to identify the contribution made by different factors it incorporates. First, we ran the system with 20 different random seeds to assess the stability of the solutions found. In each case, learning was stable and accuracy improved during training. See Figure 2 (b).

Next, we found that all of the following ablations significantly hurt reconstruction: using a flat tree (in which all languages are equidistant from the reconstructed root and from each other) instead of the consensus tree, dropping the markedness features, dropping the faithfulness features, and disabling sharing across branches. The results of these experiments are shown in Table 1.

For comparison, we also included in the same table the performance of a semi-supervised system trained by K-fold validation. The system was ran $K = 5$ times, with $1 - K^{-1}$ of the POc words given to the system as observations in the graphical model for each run. It is semi-supervised in the sense that gold reconstruction for many internal nodes are not available in the dataset (for example the common ancestor of Kwara'ae (Kw.) and Lau in Figure 3 (b)), so they are still not filled.[3]

Figure 3 (b) shows the results of a concrete run over 32 languages, zooming in to a pair of the Solomonic languages and the cognate set from Figure 1 (a). In the example shown, the reconstruction is as good as the ORACLE (described in Section 5.2), though off by one character (the final /s/ is not present in any of the 32 inputs and therefore is not reconstructed). In (a), diagrams show, for both the global and the local (Kwara'ae) features, the expectations of each substitution superimposed on an IPA sound chart, as well as a list of the top changes. Darker lines indicate higher counts. This run did not use natural class constraints, but it can

[3]We also tried a fully supervised system where a flat topology is used so that all of these latent internal nodes are avoided; but it did not perform as well—this is consistent with the -Topology experiment of Table 1.

be seen that linguistically plausible substitutions are learned. The global features prefer a range of voicing changes, manner changes, adjacent vowel motion, and so on, including mutations like /s/ to /h/ which are common but poorly represented in a naive attribute-based natural class scheme. On the other hand, the features local to the language Kwara'ae pick out the subset of these changes which are active in that branch, such as /s/→/t/ fortition.

5.2 Comparisons against other methods

The first two competing methods, PRAGUE and BCLKG, are described in Oakes (2000) and Bouchard-Côté et al. (2008) respectively and summarized in Section 1. Neither approach scales well to large datasets. In the first case, the bottleneck is the complexity of computing multi-alignments without guide trees and the vanishing probability that independent reconstructions agree. In the second case, the problem comes from the unregularized proliferation of parameters and slow mixing of the inference algorithm. For this reason, we built a third baseline that scales well in large datasets.

This third baseline, CENTROID, computes the centroid of the observed word forms in Levenshtein distance. Let $L(x, y)$ denote the Levenshtein distance between word forms x and y. Ideally, we would like the baseline to return $\text{argmin}_{x \in \Sigma^*} \sum_{y \in O} L(x, y)$, where $O = \{y_1, \ldots, y_{|O|}\}$ is the set of observed word forms. Note that the optimum is not changed if we restrict the minimization to be taken on $x \in \Sigma(O)^*$ such that $m \leq |x| \leq M$ where $m = \min_i |y_i|, M = \max_i |y_i|$ and $\Sigma(O)$ is the set of characters occurring in O. Even with this restriction, this optimization is intractable. As an approximation, we considered only strings built by at most k contiguous substrings taken from the word forms in O. If $k = 1$, then it is equivalent to taking the min over $x \in O$. At the other end of the spectrum, if $k = M$, it is exact. This scheme is exponential in k, but since words are relatively short, we found that $k = 2$ often finds the same solution as higher values of k. The difference was in all the cases not statistically significant, so we report the approximation $k = 2$ in what follows.

We also compared against an oracle, denoted ORACLE, which returns $\text{argmin}_{y \in O} L(y, x^*)$, where x^* is the target reconstruction. We will denote it by ORACLE. This is superior to picking a single closest language to be used for all word forms, but it is possible for systems to perform better than the oracle since it has to return one of the observed word forms.

Comparison	CENTROID	PRAGUE	BCLKG
Protolanguage	POc	PMJ	La
Heldout (prop.)	243 (1.0)	79 (1.0)	293 (0.5)
Modern languages	70	4	2
Cognate sets	1321	179	583
Observed words	10783	470	1463
Mean word length	4.5	5.0	7.4

Table 2: Experimental setup: number of held-out protoword from (absolute and relative), of modern languages, cognate sets and total observed words. The split for BCLKG is the same as in Bouchard-Côté et al. (2008).

We performed the comparison against Oakes (2000) and Bouchard-Côté et al. (2008) on the same dataset and experimental conditions as those used in the respective papers (see Table 2). Note that the setup of Bouchard-Côté et al. (2008) provides supervision (half of the Latin word forms are provided); all of the other comparisons are performed in a completely unsupervised manner.

The PMJ dataset was compiled by Nothofer (1975), who also reconstructed the corresponding protolanguage. Since PRAGUE is not guaranteed to return a reconstruction for each cognate set, only 55 word forms could be directly compared to our system. We restricted comparison to this subset of the data. This favors PRAGUE since the system only proposes a reconstruction when it is certain. Still, our system outperformed PRAGUE, with an average distance of 1.60 compared to 2.02 for PRAGUE. The difference is marginally significant, $p = 0.06$, partly due to the small number of word forms involved.

We also exceeded the performance of BCLKG on the Romance dataset. Our system's reconstruction had an edit distance of 3.02 to the truth against 3.10 for BCLKG. However, this difference was not significant ($p = 0.15$). We think this is because of the high level of noise in the data (the Romance dataset is the only dataset we consider that was automatically constructed rather than curated by linguists). A second factor contributing to this small difference may be that the the experimental setup of BCLKG used very few languages, while the performance of our system improves markedly with more languages.

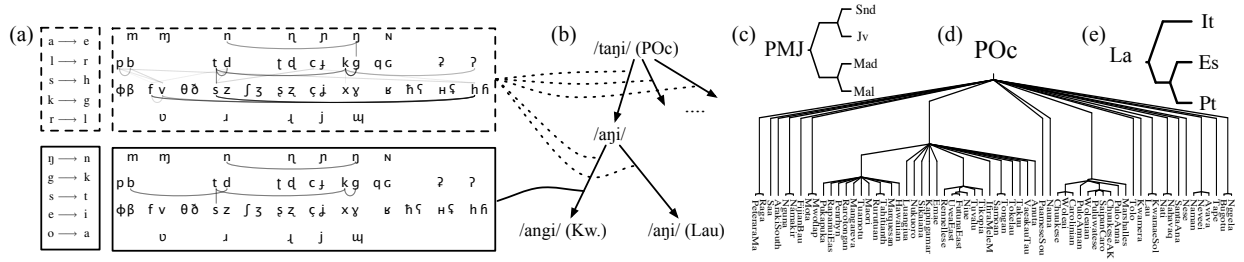


Figure 3: (a) A visualization of two learned faithfulness parameters: on the top, from the universal features, on the bottom, for one particular branch. Each pair of phonemes have a link with grayscale value proportional to the expectation of a transition between them. The five strongest links are also included at the right. (b) A sample taken from our POc experiments (see text). (c-e) Phylogenetic trees for three language families: Proto-Malayo-Javanic, Austronesian and Romance.

We conducted another experiment to verify this by running both systems in larger trees. Because the Romance dataset had only three modern languages transcribed in IPA, we used the Austronesian dataset to perform the test. The results were all significant in this setup: while our method went from an edit distance of 2.01 to 1.79 in the 4-to-8 languages experiment described in Section 5.1, BCLKG went from 3.30 to 3.38. This suggests that more languages can actually *hurt* systems that do not support parameter sharing.

Since we have shown evidence that PRAGUE and BCLKG do not scale well to large datasets, we also compared against ORACLE and CENTROID in a large-scale setting. Specifically, we compare to the experimental setup on 64 modern languages used to reconstruct POc described before. Encouragingly, while the system's average distance (1.49) does not attain that of the ORACLE (1.13), we significantly outperform the CENTROID baseline (1.79).

5.3 Incorporating prior linguistic knowledge

The model also supports the addition of prior linguistic knowledge. This takes the form of feature templates with more internal structure. We performed experiments with an additional feature template:

STRUCT-FAITHFULNESS is a structured version of FAITHFULNESS, replacing x and y with their natural classes $N_\beta(x)$ and $N_\beta(y)$ where β indexes types of classes, ranging over {manner, place, phonation, isOral, isCentral, height, backness, roundedness}. This feature set is reminiscent of the featurized representation of Kondrak (2000).

We compared the performance of the system with and without STRUCT-FAITHFULNESS to check if the algorithm can recover the structure of natural classes in an unsupervised fashion. We found that with 2 or 4 observed languages, FAITHFULNESS underperformed STRUCT-FAITHFULNESS, but for larger trees, the difference was not significant. FAITHFULNESS even slightly outperformed its structured cousin with 16 observed languages.

6 Conclusion

By enriching our model to include important features like markedness, and by scaling up to much larger data sets than were previously possible, we obtained substantial improvements in reconstruction quality, giving the best results on past data sets. While many more complex phenomena are still unmodeled, from reduplication to borrowing to chained sound shifts, the current approach significantly increases the power, accuracy, and efficiency of automatic reconstruction.

Acknowledgments

We would like to thank Anna Rafferty and our reviewers for their comments. This work was supported by a NSERC fellowship to the first author and NSF grant number BCS-0631518 to the second author.

References

R. Blust. 1993. Central and central-Eastern Malayo-Polynesian. *Oceanic Linguistics*, 32:241–293.

A. Bouchard-Côté, P. Liang, D. Klein, and T. L. Griffiths. 2008. A probabilistic approach to language change. In *Advances in Neural Information Processing Systems 20*.

A. Bouchard-Côté, M. I. Jordan, and D. Klein. 2009. Efficient inference in phylogenetic InDel trees. In *Advances in Neural Information Processing Systems 21*.

L. Campbell. 1998. *Historical Linguistics*. The MIT Press.

S. F. Chen. 2003. Conditional and joint models for grapheme-to-phoneme conversion. In *Proceedings of Eurospeech*.

M. A. Covington. 1998. Alignment of multiple languages for historical comparison. In *Proceedings of ACL 1998*.

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. Series B (Methodological)*, 39(1):1–38.

M. Dreyer, J. R. Smith, and J. Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of EMNLP 2008*.

S. P. Durham and D. E. Rogers. 1969. An application of computer programming to the reconstruction of a proto-language. In *Proceedings of the 1969 conference on Computational linguistics*.

C. L. Eastlack. 1977. Iberochange: A program to simulate systematic sound change in Ibero-Romance. *Computers and the Humanities*.

J. Felsenstein. 1989. PHYLIP - PHYLogeny Inference Package (Version 3.2). *Cladistics*, 5:164–166.

S. Goldwater and M. Johnson. 2003. Learning OT constraint rankings using a maximum entropy model. *Proceedings of the Workshop on Variation within Optimality Theory*.

S. J. Greenhill, R. Blust, and R. D. Gray. 2008. The Austronesian basic vocabulary database: From bioinformatics to lexomics. *Evolutionary Bioinformatics*, 4:271–283.

H. H. Hock. 1986. *Principles of Historical Linguistics*. Walter de Gruyter.

I. Holmes and W. J. Bruno. 2001. Evolutionary HMM: a Bayesian approach to multiple alignment. *Bioinformatics*, 17:803–820.

W. Jank. 2005. Stochastic variants of EM: Monte Carlo, quasi-Monte Carlo and more. In *Proceedings of the American Statistical Association*.

G. Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proceedings of NAACL 2000*.

G. Kondrak. 2002. *Algorithms for Language Reconstruction*. Ph.D. thesis, University of Toronto.

V. I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10, February.

D. C. Liu, J. Nocedal, and C. Dong. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528.

J. B. Lowe and M. Mazaudon. 1994. The reconstruction engine: a computer implementation of the comparative method. *Comput. Linguist.*, 20(3):381–417.

G. A. Lunter, I. Miklós, Y. S. Song, and J. Hein. 2003. An efficient algorithm for statistical multiple alignment on arbitrary phylogenetic trees. *Journal of Computational Biology*, 10:869–889.

B. Nothofer. 1975. *The reconstruction of Proto-Malayo-Javanic*. M. Nijhoff.

M. P. Oakes. 2000. Computer estimation of vocabulary in a protolanguage from word lists in four daughter languages. *Journal of Quantitative Linguistics*, 7(3):233–244.

A. Prince and P. Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. Technical Report 2, Rutgers University Center for Cognitive Science.

L. Tierney. 1994. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22(4):1701–1728.

A. Varadarajan, R. K. Bradley, and I. H. Holmes. 2008. Tools for simulating evolution of aligned genomic regions with integrated parameter estimation. *Genome Biology*, 9:R147.

C. Wilson. 2006. Learning phonology with substantive bias: An experimental and computational study of velar palatalization. *Cognitive Science*, 30.5:945–982.

Shared Logistic Normal Distributions for Soft Parameter Tying in Unsupervised Grammar Induction

Shay B. Cohen and **Noah A. Smith**
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{scohen,nasmith}@cs.cmu.edu

Abstract

We present a family of priors over probabilistic grammar weights, called the shared logistic normal distribution. This family extends the partitioned logistic normal distribution, enabling factored covariance between the probabilities of different derivation events in the probabilistic grammar, providing a new way to encode prior knowledge about an unknown grammar. We describe a variational EM algorithm for learning a probabilistic grammar based on this family of priors. We then experiment with unsupervised dependency grammar induction and show significant improvements using our model for both monolingual learning and *bilingual* learning with a non-parallel, multilingual corpus.

1 Introduction

Probabilistic grammars have become an important tool in natural language processing. They are most commonly used for parsing and linguistic analysis (Charniak and Johnson, 2005; Collins, 2003), but are now commonly seen in applications like machine translation (Wu, 1997) and question answering (Wang et al., 2007). An attractive property of probabilistic grammars is that they permit the use of well-understood parameter estimation methods for *learning*—both from labeled and unlabeled data. Here we tackle the unsupervised grammar learning problem, specifically for unlexicalized context-free dependency grammars, using an empirical Bayesian approach with a novel family of priors.

There has been an increased interest recently in employing Bayesian modeling for probabilistic grammars in different settings, ranging from putting priors over grammar probabilities (Johnson et al., 2007) to putting non-parametric priors over derivations (Johnson et al., 2006) to learning the set of states in a grammar (Finkel et al., 2007; Liang et al., 2007). Bayesian methods offer an elegant framework for combining prior knowledge with data. The main challenge in Bayesian grammar learning is efficiently approximating probabilistic inference, which is generally intractable. Most commonly variational (Johnson, 2007; Kurihara and Sato, 2006) or sampling techniques are applied (Johnson et al., 2006).

Because probabilistic grammars are built out of multinomial distributions, the Dirichlet family (or, more precisely, a collection of Dirichlets) is a natural candidate for probabilistic grammars because of its conjugacy to the multinomial family. Conjugacy implies a clean form for the posterior distribution over grammar probabilities (given the data and the prior), bestowing computational tractability.

Following work by Blei and Lafferty (2006) for topic models, Cohen et al. (2008) proposed an alternative to Dirichlet priors for probabilistic grammars, based on the logistic normal (LN) distribution over the probability simplex. Cohen et al. used this prior to softly tie grammar weights through the covariance parameters of the LN. The prior encodes information about which grammar rules' weights are likely to covary, a more intuitive and expressive representation of knowledge than offered by Dirichlet distributions.[1]

The contribution of this paper is two-fold. First, from the modeling perspective, we present a generalization of the LN prior of Cohen et al. (2008), showing how to extend the use of the LN prior to

[1] Although the task, underlying model, and weights being tied were different, Eisner (2002) also showed evidence for the efficacy of parameter tying in grammar learning.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 74–82,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

tie between *any* grammar weights in a probabilistic grammar (instead of only allowing weights within the same multinomial distribution to covary). Second, from the experimental perspective, we show how such flexibility in parameter tying can help in unsupervised grammar learning in the well-known monolingual setting and in a new *bilingual* setting where grammars for two languages are learned at once (without parallel corpora).

Our method is based on a distribution which we call the **shared logistic normal distribution**, which is a distribution over a collection of multinomials from different probability simplexes. We provide a variational EM algorithm for inference.

The rest of this paper is organized as follows. In §2, we give a brief explanation of probabilistic grammars and introduce some notation for the specific type of dependency grammar used in this paper, due to Klein and Manning (2004). In §3, we present our model and a variational inference algorithm for it. In §4, we report on experiments for both monolingual settings and a bilingual setting and discuss them. We discuss future work (§5) and conclude in §6.

2 Probabilistic Grammars and Dependency Grammar Induction

A probabilistic grammar defines a probability distribution over grammatical derivations generated through a step-by-step process. HMMs, for example, can be understood as a random walk through a probabilistic finite-state network, with an output symbol sampled at each state. Each "step" of the walk and each symbol emission corresponds to one derivation step. PCFGs generate phrase-structure trees by recursively rewriting nonterminal symbols as sequences of "child" symbols (each itself either a nonterminal symbol or a terminal symbol analogous to the emissions of an HMM). Each step or emission of an HMM and each rewriting operation of a PCFG is conditionally independent of the other rewriting operations given a single structural element (one HMM or PCFG state); this Markov property permits efficient inference for the probability distribution defined by the probabilistic grammar.

In general, a probabilistic grammar defines the joint probability of a string $\mathbf{x}$ and a grammatical derivation $\mathbf{y}$:

$$p(\mathbf{x}, \mathbf{y} \mid \boldsymbol{\theta}) = \prod_{k=1}^{K} \prod_{i=1}^{N_k} \theta_{k,i}^{f_{k,i}(\mathbf{x},\mathbf{y})} \quad (1)$$
$$= \exp \sum_{k=1}^{K} \sum_{i=1}^{N_k} f_{k,i}(\mathbf{x}, \mathbf{y}) \log \theta_{k,i}$$

where $f_{k,i}$ is a function that "counts" the number of times the kth distribution's ith event occurs in the derivation. The $\boldsymbol{\theta}$ are a collection of K multinomials $\langle \boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_K \rangle$, the kth of which includes N_k events. Note that there may be many derivations $\mathbf{y}$ for a given string $\mathbf{x}$—perhaps even infinitely many in some kinds of grammars.

2.1 Dependency Model with Valence

HMMs and PCFGs are the best-known probabilistic grammars, but there are many others. In this paper, we use the "dependency model with valence" (DMV), due to Klein and Manning (2004). DMV defines a probabilistic grammar for unlabeled, projective dependency structures. Klein and Manning (2004) achieved their best results with a combination of DMV with a model known as the "constituent-context model" (CCM). We do not experiment with CCM in this paper, because it does not fit directly in a Bayesian setting (it is highly deficient) and because state-of-the-art unsupervised dependency parsing results have been achieved with DMV alone (Smith, 2006).

Using the notation above, DMV defines $\mathbf{x} = \langle x_1, x_2, ..., x_n \rangle$ to be a sentence. x_0 is a special "wall" symbol, \$, on the left of every sentence. A tree $\mathbf{y}$ is defined by a pair of functions $\mathbf{y}_{left}$ and $\mathbf{y}_{right}$ (both $\{0, 1, 2, ..., n\} \rightarrow 2^{\{1,2,...,n\}}$) that map each word to its sets of left and right dependents, respectively. Here, the graph is constrained to be a *projective* tree rooted at $x_0 = \$$: each word except \$ has a single parent, and there are no cycles or crossing dependencies. $\mathbf{y}_{left}(0)$ is taken to be empty, and $\mathbf{y}_{right}(0)$ contains the sentence's single head. Let $\mathbf{y}^{(i)}$ denote the subtree rooted at position i. The probability $P(\mathbf{y}^{(i)} \mid x_i, \boldsymbol{\theta})$ of generating this subtree, given its head word x_i, is defined recursively, as described in Fig. 1 (Eq. 2).

The probability of the entire tree is given by $p(\mathbf{x}, \mathbf{y} \mid \boldsymbol{\theta}) = P(\mathbf{y}^{(0)} \mid \$, \boldsymbol{\theta})$. The $\boldsymbol{\theta}$ are the multinomial distributions $\theta_{\mathrm{s}}(\cdot \mid \cdot, \cdot, \cdot)$ and $\theta_{\mathrm{c}}(\cdot \mid \cdot, \cdot)$. To

$$P(\mathbf{y}^{(i)} \mid x_i, \boldsymbol{\theta}) = \prod_{D \in \{left, right\}} \theta_{\mathrm{s}}(\mathrm{stop} \mid x_i, D, [\mathbf{y}_D(i) = \emptyset]) \times \prod_{j \in \mathbf{y}_D(i)} \theta_{\mathrm{s}}(\neg\mathrm{stop} \mid x_i, D, \mathrm{first}_{\mathbf{y}}(j)) \times \theta_{\mathrm{c}}(x_j \mid x_i, D) \times P(\mathbf{y}^{(j)} \mid x_j, \boldsymbol{\theta}) \quad (2)$$

Figure 1: The "dependency model with valence" recursive equation. $\mathrm{first}_{\mathbf{y}}(j)$ is a predicate defined to be true iff x_j is the closest child (on either side) to its parent x_i. The probability of the tree $p(\mathbf{x}, \mathbf{y} \mid \boldsymbol{\theta}) = P(\mathbf{y}^{(0)} \mid \$, \boldsymbol{\theta})$.

follow the general setting of Eq. 1, we index these distributions as $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_K$.

Headden et al. (2009) extended DMV so that the distributions θ_c condition on the valence as well, with smoothing, and showed significant improvements for short sentences. Our experiments found that these improvements do not hold on longer sentences. Here we experiment only with DMV, but note that our techniques are also applicable to richer probabilistic grammars like that of Headden et al.

2.2 Learning DMV

Klein and Manning (2004) learned the DMV probabilities $\boldsymbol{\theta}$ from a corpus of part-of-speech-tagged sentences using the EM algorithm. EM manipulates $\boldsymbol{\theta}$ to locally optimize the likelihood of the observed portion of the data (here, $\mathbf{x}$), marginalizing out the hidden portions (here, $\mathbf{y}$). The likelihood surface is not globally concave, so EM only locally optimizes the surface. Klein and Manning's initialization, though reasonable and language-independent, was an important factor in performance.

Various alternatives to EM were explored by Smith (2006), achieving substantially more accurate parsing models by altering the objective function. Smith's methods did require substantial hyperparameter tuning, and the best results were obtained using small annotated development sets to choose hyperparameters. In this paper, we consider only fully unsupervised methods, though we the Bayesian ideas explored here might be merged with the biasing approaches of Smith (2006) for further benefit.

3 Parameter Tying in the Bayesian Setting

As stated above, $\boldsymbol{\theta}$ comprises a collection of multinomials that weights the grammar. Taking the Bayesian approach, we wish to place a prior on those multinomials, and the Dirichlet family is a natural candidate for such a prior because of its conjugacy, which makes inference algorithms easier to derive. For example, if we make a "mean-field assumption," with respect to hidden structure and weights, the variational algorithm for approximately inferring the distribution over $\boldsymbol{\theta}$ and trees $\mathbf{y}$ resembles the traditional EM algorithm very closely (Johnson, 2007). In fact, variational inference in this case takes an action similar to smoothing the counts using the exp-Ψ function during the E-step. Variational inference can be embedded in an empirical Bayes setting, in which we optimize the variational bound with respect to the hyperparameters as well, repeating the process until convergence.

3.1 Logistic Normal Distributions

While Dirichlet priors over grammar probabilities make learning algorithms easy, they are limiting. In particular, as noted by Blei and Lafferty (2006), there is no explicit flexible way for the Dirichlet's parameters to encode beliefs about *covariance* between the probabilities of two events. To illustrate this point, we describe how a multinomial $\boldsymbol{\theta}$ of dimension d is generated from a Dirichlet distribution with parameters $\boldsymbol{\alpha} = \langle \alpha_1, ..., \alpha_d \rangle$:

1. Generate $\eta_j \sim \Gamma(\alpha_j, 1)$ independently for $j \in \{1, ..., d\}$.
2. $\theta_j \leftarrow \eta_j / \sum_i \eta_i$.

where $\Gamma(\alpha, 1)$ is a Gamma distribution with shape α and scale 1.

Correlation among θ_i and θ_j, $i \neq j$, cannot be modeled directly, only through the normalization in step 2. In contrast, LN distributions (Aitchison, 1986) provide a natural way to model such correlation. The LN draws a multinomial $\boldsymbol{\theta}$ as follows:

1. Generate $\boldsymbol{\eta} \sim \mathrm{Normal}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.
2. $\theta_j \leftarrow \exp(\eta_j) / \sum_i \exp(\eta_i)$.

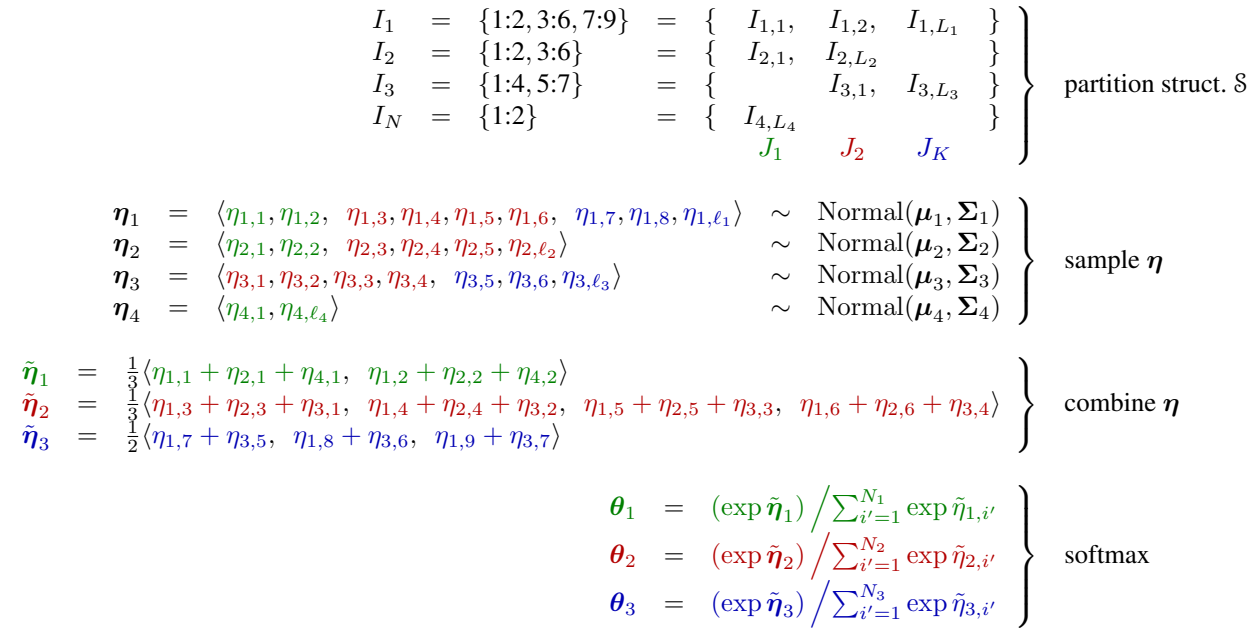


Figure 2: An example of a shared logistic normal distribution, illustrating Def. 1. $N = 4$ experts are used to sample $K = 3$ multinomials; $L_1 = 3$, $L_2 = 2$, $L_3 = 2$, $L_4 = 1$, $\ell_1 = 9$, $\ell_2 = 6$, $\ell_3 = 7$, $\ell_4 = 2$, $N_1 = 2$, $N_2 = 4$, and $N_3 = 3$. This figure is best viewed in color.

Blei and Lafferty (2006) defined correlated topic models by replacing the Dirichlet in latent Dirichlet allocation models (Blei et al., 2003) with a LN distribution. Cohen et al. (2008) compared Dirichlet and LN distributions for learning DMV using empirical Bayes, finding substantial improvements for English using the latter.

In that work, we obtained improvements even without specifying exactly *which* grammar probabilities covaried. While empirical Bayes learning permits these covariances to be discovered without supervision, we found that by initializing the covariance to encode beliefs about which grammar probabilities should covary, further improvements were possible. Specifically, we grouped the Penn Treebank part-of-speech tags into coarse groups based on the treebank annotation guidelines and biased the initial covariance matrix for each child distribution $\theta_c(\cdot \mid \cdot, \cdot)$ so that the probabilities of child tags from the same coarse group covaried. For example, the probability that a past-tense verb (VBD) has a singular noun (NN) as a right child may be correlated with the probability that it has a *plural* noun (NNS) as a right child. Hence linguistic knowledge—specifically, a coarse grouping of word classes—can be encoded in the prior.

A per-distribution LN distribution only permits probabilities within a multinomial to covary. We will generalize the LN to permit covariance among any probabilities in $\boldsymbol{\theta}$, throughout the model. For example, the probability of a past-tense verb (VBD) having a noun as a right child might correlate with the probability that other kinds of verbs (VBZ, VBN, etc.) have a noun as a right child.

The *partitioned logistic normal distribution* (PLN) is a generalization of the LN distribution that takes the first step towards our goal (Aitchison, 1986). Generating from PLN involves drawing a random vector from a multivariate normal distribution, but the logistic transformation is applied to different parts of the vector, leading to sampled multinomial distributions of the required lengths from different probability simplices. This is in principle what is required for arbitrary covariance between grammar probabilities, except that DMV has $O(t^2)$ weights for a part-of-speech vocabulary of size t, requiring a very large multivariate normal distribution with $O(t^4)$ covariance parameters.

3.2 Shared Logistic Normal Distributions

To solve this problem, we suggest a refinement of the class of PLN distributions. Instead of using a single normal vector for all of the multinomials, we use several normal vectors, partition each one and then *recombine* parts which correspond to the same multinomial, as a mixture. Next, we apply the logisitic transformation on the mixed vectors (each of which is normally distributed as well). Fig. 2 gives an example of a non-trivial case of using a SLN distribution, where three multinomials are generated from four normal experts.

We now formalize this notion. For a natural number N, we denote by $1{:}N$ the set $\{1, ..., N\}$. For a vector in $v \in \mathbb{R}^N$ and a set $I \subseteq 1{:}N$, we denote by v_I to be the vector created from v by using the coordinates in I. Recall that K is the number of multinomials in the probabilistic grammar, and N_k is the number of events in the kth multinomial.

Definition 1. *We define a shared logistic normal distribution with N "experts" over a collection of K multinomial distributions. Let $\boldsymbol{\eta}_n \sim \mathrm{Normal}(\boldsymbol{\mu}_n, \Sigma_n)$ be a set of multivariate normal variables for $n \in 1{:}N$, where the length of $\boldsymbol{\eta}_n$ is denoted ℓ_n. Let $I_n = \{I_{n,j}\}_{j=1}^{L_n}$ be a partition of $1{:}\ell_n$ into L_n sets, such that $\cup_{j=1}^{L_n} I_{n,j} = 1{:}\ell_n$ and $I_{n,j} \cap I_{n,j'} = \emptyset$ for $j \neq j'$. Let J_k for $k \in 1{:}K$ be a collection of (disjoint) subsets of $\{I_{n,j} \mid n \in 1{:}N, j \in 1{:}\ell_n, |I_{n,j}| = N_k\}$, such that all sets in J_k are of the same size, N_k. Let $\tilde{\boldsymbol{\eta}}_k = \frac{1}{|J_k|} \sum_{I_{n,j} \in J_k} \boldsymbol{\eta}_{n, I_{n,j}}$, and $\theta_{k,i} = \exp(\tilde{\eta}_{k,i}) \,/ \sum_{i'} \exp(\tilde{\eta}_{k,i'})$. We then say $\boldsymbol{\theta}$ distributes according to the* shared logistic normal distribution *with* partition structure $\mathcal{S} = (\{I_n\}_{n=1}^N, \{J_k\}_{k=1}^K)$ *and* normal experts $\{(\boldsymbol{\mu}_n, \Sigma_n)\}_{n=1}^N$ *and denote it by* $\boldsymbol{\theta} \sim \mathrm{SLN}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathcal{S})$.

The partitioned LN distribution in Aitchison (1986) can be formulated as a shared LN distribution where $N = 1$. The LN collection used by Cohen et al. (2008) is the special case where $N = K$, each $L_n = 1$, each $\ell_k = N_k$, and each $J_k = \{I_{k,1}\}$.

The covariance among arbitrary $\theta_{k,i}$ is not defined directly; it is implied by the definition of the normal experts $\boldsymbol{\eta}_{n, I_{n,j}}$, for each $I_{n,j} \in J_k$. We note that a SLN can be represented as a PLN by relying on the distributivity of the covariance operator, and merging all the partition structure into one (perhaps sparse) covariance matrix. However, if we are interested in keeping a factored structure on the covariance matrices which generate the grammar weights, we cannot represent every SLN as a PLN.

It is convenient to think of each $\eta_{i,j}$ as a weight associated with a unique event's probability, a certain outcome of a certain multinomial in the probabilistic grammar. By letting different $\eta_{i,j}$ covary with each other, we loosen the relationships among $\theta_{k,j}$ and permit the model—at least in principle—to learn patterns from the data. Def. 1 also implies that we multiply several multinomials together in a product-of-experts style (Hinton, 1999), because the exponential of a mixture of normals becomes a product of (unnormalized) probabilities.

Our extension to the model in Cohen et al. (2008) follows naturally after we have defined the shared LN distribution. The generative story for this model is as follows:

1. Generate $\boldsymbol{\theta} \sim \mathrm{SLN}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathcal{S})$, where $\boldsymbol{\theta}$ is a collection of vectors $\boldsymbol{\theta}_k$, $k = 1, ..., K$.
2. Generate $\mathbf{x}$ and $\mathbf{y}$ from $p(\mathbf{x}, \mathbf{y} \mid \boldsymbol{\theta})$ (i.e., sample from the probabilistic grammar).

3.3 Inference

In this work, the partition structure $\mathcal{S}$ is *known*, the sentences $\mathbf{x}$ are *observed*, the trees $\mathbf{y}$ and the grammar weights $\boldsymbol{\theta}$ are *hidden*, and the parameters of the shared LN distribution $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are *learned*.[2]

Our inference algorithm aims to find the posterior over the grammar probabilities $\boldsymbol{\theta}$ and the hidden structures (grammar trees $\mathbf{y}$). To do that, we use variational approximation techniques (Jordan et al., 1999), which treat the problem of finding the posterior as an optimization problem aimed to find the best approximation $q(\boldsymbol{\theta}, \mathbf{y})$ of the posterior $p(\boldsymbol{\theta}, \mathbf{y} \mid \mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathcal{S})$. The posterior q needs to be constrained to be within a family of tractable and manageable distributions, yet rich enough to represent good approximations of the true posterior. "Best approximation" is defined as the KL divergence between $q(\boldsymbol{\theta}, \mathbf{y})$ and $p(\boldsymbol{\theta}, \mathbf{y} \mid \mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathcal{S})$.

Our variational inference algorithm uses a mean-field assumption: $q(\boldsymbol{\theta}, \mathbf{y}) = q(\boldsymbol{\theta})q(\mathbf{y})$. The distribution $q(\boldsymbol{\theta})$ is assumed to be a LN distribution with

[2] In future work, we might aim to learn $\mathcal{S}$.

$$\log p(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathcal{S}) \geq \underbrace{\left(\sum_{n=1}^{N} \mathbb{E}_q\left[\log p(\boldsymbol{\eta}_k \mid \boldsymbol{\mu}_k, \Sigma_k)\right]\right) + \left(\sum_{k=1}^{K}\sum_{i=1}^{N_k} \tilde{f}_{k,i}\tilde{\psi}_{k,i}\right) + H(q)}_{B} \quad (3)$$

$$\tilde{f}_{k,i} \triangleq \sum_{\mathbf{y}} q(\mathbf{y}) f_{k,i}(\mathbf{x}, \mathbf{y}) \quad (4)$$

$$\tilde{\psi}_{k,i} \triangleq \tilde{\mu}^C_{k,i} - \log \tilde{\zeta}_k + 1 - \frac{1}{\tilde{\zeta}_k}\sum_{i'=1}^{N_k} \exp\left(\tilde{\mu}^C_{k,i} + \frac{(\tilde{\sigma}^C_{k,i})^2}{2}\right) \quad (5)$$

$$\tilde{\mu}^C_k \triangleq \frac{1}{|J_k|}\sum_{I_{r,j} \in J_k} \tilde{\mu}_{r,I_{r,j}} \quad (6)$$

$$(\tilde{\sigma}^C_k)^2 \triangleq \frac{1}{|J_k|^2}\sum_{I_{r,j} \in J_k} \tilde{\sigma}^2_{r,I_{r,j}} \quad (7)$$

Figure 3: Variational inference bound. Eq. 3 is the bound itself, using notation defined in Eqs. 4–7 for clarity. Eq. 4 defines expected counts of the grammar events under the variational distribution $q(\mathbf{y})$, calculated using dynamic programming. Eq. 5 describes the weights for the weighted grammar defined by $q(\mathbf{y})$. Eq. 6 and Eq. 7 describe the mean and the variance, respectively, for the multivariate normal eventually used with the weighted grammar. These values are based on the parameterization of $q(\theta)$ by $\tilde{\mu}_{i,j}$ and $\tilde{\sigma}^2_{i,j}$. An additional set of variational parameters is $\tilde{\zeta}_k$, which helps resolve the non-conjugacy of the LN distribution through a first order Taylor approximation.

all off-diagonal covariances fixed at zero (i.e., the variational parameters consist of a single mean $\tilde{\mu}_{k,i}$ and a single variance $\tilde{\sigma}^2_{k,i}$ for each $\theta_{k,i}$). There is an additional variational parameter, $\tilde{\zeta}_k$ per multinomial, which is the result of an additional variational approximation because of the lack of conjugacy of the LN distribution to the multinomial distribution. The distribution $q(\mathbf{y})$ is assumed to be defined by a DMV with unnormalized probabilities $\tilde{\boldsymbol{\psi}}$.

Inference optimizes the bound B given in Fig. 3 (Eq. 3) with respect to the variational parameters. Our variational inference algorithm is derived similarly to that of Cohen et al. (2008). Because we wish to *learn* the values of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we embed variational inference as the E step within a variational EM algorithm, shown schematically in Fig. 4. In our experiments, we use this variational EM algorithm on a training set, and then use the normal experts' means to get a point estimate for $\boldsymbol{\theta}$, the grammar weights. This is called **empirical Bayesian estimation**. Our approach differs from maximum *a posteriori* (MAP) estimation, since we re-estimate the parameters of the normal experts. Exact MAP estimation is probably not feasible; a variational algorithm like ours might be applied, though better performance is expected from adjusting the SLN to fit the data.

4 Experiments

Our experiments involve data from two treebanks: the *Wall Street Journal* Penn treebank (Marcus et al., 1993) and the Chinese treebank (Xue et al., 2004). In both cases, following standard practice, sentences were stripped of words and punctuation, leaving part-of-speech tags for the unsupervised induction of dependency structure. For English, we train on §2–21, tune on §22 (without using annotated data), and report final results on §23. For Chinese, we train on §1–270, use §301–1151 for development and report testing results on §271–300.[3]

To evaluate performance, we report the fraction of words whose predicted parent matches the gold standard corpus. This performance measure is also known as attachment accuracy. We considered two parsing methods after extracting a point estimate for the grammar: the most probable "Viterbi" parse ($\text{argmax}_{\mathbf{y}}\, p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta})$) and the minimum Bayes risk (MBR) parse ($\text{argmin}_{\mathbf{y}}\, \mathbb{E}_{p(\mathbf{y}' \mid \mathbf{x}, \boldsymbol{\theta})}[\ell(\mathbf{y}; \mathbf{x}, \mathbf{y}')]$) with dependency attachment error as the loss function (Goodman, 1996). Performance with MBR parsing is consistently higher than its Viterbi counterpart, so we report only performance with MBR parsing.

4.1 Nouns, Verbs, and Adjectives

In this paper, we use a few simple heuristics to decide which partition structure $\mathcal{S}$ to use. Our heuris-

[3] Unsupervised training for these datasets can be costly, and requires iteratively running a cubic-time inside-outside dynamic programming algorithm, so we follow Klein and Manning (2004) in restricting the training set to sentences of ten or fewer words in length. Short sentences are also less structurally ambiguous and may therefore be easier to learn from.

Input: initial parameters $\boldsymbol{\mu}^{(0)}$, $\boldsymbol{\Sigma}^{(0)}$, partition structure $\mathbb{S}$, observed data $\mathbf{x}$, number of iterations T
Output: learned parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$
$t \leftarrow 1$;
while $t \leq T$ **do**
  E-step (for $\ell = 1, ..., M$*) do*: **repeat**
    optimize B w.r.t. $\tilde{\mu}_r^{\ell,(t)}, r = 1, ..., N$;
    optimize B w.r.t. $\tilde{\sigma}_r^{\ell,(t)}, r = 1, ..., N$;
    update $\tilde{\zeta}_r^{\ell,(t)}, r = 1, ..., N$;
    update $\tilde{\boldsymbol{\psi}}_r^{\ell,(t)}, r = 1, ..., N$;
    compute counts $\tilde{\mathbf{f}}_r^{\ell,(t)}, r = 1, ..., N$;
  until *convergence of* B ;
  M-step: optimize B w.r.t. $\boldsymbol{\mu}^{(t)}$ and $\boldsymbol{\Sigma}^{(t)}$;
  $t \leftarrow t + 1$;
end
return $\boldsymbol{\mu}^{(T)}$, $\boldsymbol{\Sigma}^{(T)}$

Figure 4: Main details of the variational inference EM algorithm with empirical Bayes estimation of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. B is the bound defined in Fig. 3 (Eq. 3). N is the number of normal experts for the SLN distribution defining the prior. M is the number of training examples. The full algorithm is given in Cohen and Smith (2009).

tics rely mainly on the centrality of content words: nouns, verbs, and adjectives. For example, in the English treebank, the most common attachment errors (with the LN prior from Cohen et al., 2008) happen with a noun (25.9%) or a verb (16.9%) parent. In the Chinese treebank, the most common attachment errors happen with noun (36.0%) and verb (21.2%) parents as well. The errors being governed by such attachments are the direct result of nouns and verbs being the most common parents in these data sets.

Following this observation, we compare four different settings in our experiments (all SLN settings include one normal expert for each multinomial on its own, equivalent to the regular LN setting from Cohen et al.):

- TIEV: We add normal experts that tie all probabilities corresponding to a verbal parent (*any* parent, using the coarse tags of Cohen et al., 2008). Let $\boldsymbol{V}$ be the set of part-of-speech tags which belong to the verb category. For each direction D (left or right), the set of multinomials of the form $\theta_{\mathrm{c}}(\cdot \mid v, D)$, for $v \in \boldsymbol{V}$, all share a normal expert. For each direction D and each boolean value B of the predicate $\mathrm{first}_{\mathbf{y}}(\cdot)$, the set of multinomials $\theta_{\mathrm{s}}(\cdot \mid x, D, v)$, for $v \in \boldsymbol{V}$ share a normal expert.
- TIEN: This is the same as TIEV, only for nominal parents.
- TIEV&N: Tie both verbs and nouns (in separate partitions). This is equivalent to taking the union of the partition structures of the above two settings.
- TIEA: This is the same as TIEV, only for adjectival parents.

Since inference for a model with parameter tying can be computationally intensive, we first run the inference algorithm without parameter tying, and then add parameter tying to the rest of the inference algorithm's execution until convergence.

Initialization is important for the inference algorithm, because the variational bound is a non-concave function. For the expected values of the normal experts, we use the initializer from Klein and Manning (2004). For the covariance matrices, we follow the setting in Cohen et al. (2008) in our experiments also described in §3.1. For each treebank, we divide the tags into twelve disjoint tag families.[4] The covariance matrices for all dependency distributions were initialized with 1 on the diagonal, 0.5 between tags which belong to the same family, and 0 otherwise. This initializer has been shown to be more successful than an identity covariance matrix.

4.2 Monolingual Experiments

We begin our experiments with a monolingual setting, where we learn grammars for English and Chinese (separately) using the settings described above.

The attachment accuracy for this set of experiments is described in Table 1. The baselines include right attachment (where each word is attached to the word to its right), MLE via EM (Klein and Manning, 2004), and empirical Bayes with Dirichlet and LN priors (Cohen et al., 2008). We also include a "ceiling" (DMV trained using supervised MLE from the training sentences' trees). For English, we see that tying nouns, verbs or adjectives improves performance compared to the LN baseline. Tying both nouns and verbs improves performance a bit more.

[4] These are simply coarser tags: adjective, adverb, conjunction, foreign word, interjection, noun, number, particle, preposition, pronoun, proper noun, verb.

		attachment acc. (%)		
		≤ 10	≤ 20	all
English	Attach-Right	38.4	33.4	31.7
	EM (K&M, 2004)	46.1	39.9	35.9
	Dirichlet	46.1	40.6	36.9
	LN (CG&S, 2008)	59.4	45.9	40.5
	SLN, TIEV	60.2	46.2	40.0
	SLN, TIEN	60.2	46.7	40.9
	SLN, TIEV&N	61.3	47.4	41.4
	SLN, TIEA	59.9	45.8	40.9
	Biling. SLN, TIEV	†61.6	47.6	41.7
	Biling. SLN, TIEN	†61.8	**48.1**	†42.1
	Biling. SLN, TIEV&N	**62.0**	†48.0	**42.2**
	Biling. SLN, TIEA	61.3	47.6	41.7
	Supervised MLE	*84.5*	*74.9*	*68.8*
Chinese	Attach-Right	34.9	34.6	34.6
	EM (K&M, 2004)	38.3	36.1	32.7
	Dirichlet	38.3	35.9	32.4
	LN	50.1	40.5	**35.8**
	SLN, TIEV	†51.9	**42.0**	**35.8**
	SLN, TIEN	43.0	38.4	33.7
	SLN, TIEV&N	45.0	39.2	34.2
	SLN, TIEA	47.4	40.4	35.2
	Biling. SLN, TIEV	†51.9	**42.0**	**35.8**
	Biling. SLN, TIEN	48.0	38.9	33.8
	Biling. SLN, TIEV&N	†51.5	†41.7	35.3
	Biling. SLN, TIEA	**52.0**	41.3	35.2
	Supervised MLE	*84.3*	*66.1*	*57.6*

Table 1: Attachment accuracy of different models, on test data from the Penn Treebank and the Chinese Treebank of varying levels of difficulty imposed through a length filter. Attach-Right attaches each word to the word on its right and the last word to \$. Bold marks best overall accuracy per length bound, and † marks figures that are not significantly worse (binomial sign test, $p < 0.05$).

4.3 Bilingual Experiments

Leveraging information from one language for the task of disambiguating another language has received considerable attention (Dagan, 1991; Smith and Smith, 2004; Snyder and Barzilay, 2008; Burkett and Klein, 2008). Usually such a setting requires a parallel corpus or other annotated data that ties between those two languages.[5]

Our bilingual experiments use the English and Chinese treebanks, which are not parallel corpora, to train parsers for both languages jointly. Sharing information between those two models is done by softly tying grammar weights in the two hidden grammars.

We first merge the models for English and Chinese by taking a union of the multinomial families of each and the corresponding prior parameters. We then add a normal expert that ties between the parts of speech in the respective partition structures for both grammars together. Parts of speech are matched through the single coarse tagset (footnote 4). For example, with TIEV, let $\boldsymbol{V} = \boldsymbol{V}_{\text{Eng}} \cup \boldsymbol{V}_{\text{Chi}}$ be the set of part-of-speech tags which belong to the verb category for either treebank. Then, we tie parameters for all part-of-speech tags in $\boldsymbol{V}$. We tested this joint model for each of TIEV, TIEN, TIEV&N, and TIEA. After running the inference algorithm which learns the two models jointly, we use unseen data to test each learned model separately.

Table 1 includes the results for these experiments. The performance on English improved significantly in the bilingual setting, achieving highest performance with TIEV&N. Performance with Chinese is also the highest in the bilingual setting, with TIEA and TIEV&N.

5 Future Work

In future work we plan to lexicalize the model, including a Bayesian grammar prior that accounts for the syntactic patterns of *words*. Nonparametric models (Teh, 2006) may be appropriate. We also believe that Bayesian discovery of cross-linguistic patterns is an exciting topic worthy of further exploration.

6 Conclusion

We described a Bayesian model that allows soft parameter tying among *any* weights in a probabilistic grammar. We used this model to improve unsupervised parsing accuracy on two different languages, English and Chinese, achieving state-of-the-art results. We also showed how our model can be effectively used to simultaneously learn grammars in two languages from non-parallel multilingual data.

Acknowledgments

This research was supported by NSF IIS-0836431. The authors thank the anonymous reviewers and Sylvia Rebholz for helpful comments.

[5] Haghighi et al. (2008) presented a technique to learn bilingual lexicons from two non-parallel monolingual corpora.

References

J. Aitchison. 1986. *The Statistical Analysis of Compositional Data*. Chapman and Hall, London.

D. M. Blei and J. D. Lafferty. 2006. Correlated topic models. In *Proc. of NIPS*.

D. M. Blei, A. Ng, and M. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

D. Burkett and D. Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proc. of EMNLP*.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*.

S. B. Cohen and N. A. Smith. 2009. Inference for probabilistic grammars with shared logistic normal distributions. Technical report, Carnegie Mellon University.

S. B. Cohen, K. Gimpel, and N. A. Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *NIPS*.

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

I. Dagan. 1991. Two languages are more informative than one. In *Proc. of ACL*.

J. Eisner. 2002. Transformational priors over grammars. In *Proc. of EMNLP*.

J. R. Finkel, T. Grenager, and C. D. Manning. 2007. The infinite tree. In *Proc. of ACL*.

J. Goodman. 1996. Parsing algorithms and metrics. In *Proc. of ACL*.

A. Haghighi, P. Liang, T. Berg-Kirkpatrick, and D. Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proc. of ACL*.

W. P. Headden, M. Johnson, and D. McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proc. of NAACL-HLT*.

G. E. Hinton. 1999. Products of experts. In *Proc. of ICANN*.

M. Johnson, T. L. Griffiths, and S. Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparameteric Bayesian models. In *NIPS*.

M. Johnson, T. L. Griffiths, and S. Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proc. of NAACL*.

M. Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proc. EMNLP-CoNLL*.

M. I. Jordan, Z. Ghahramani, T. S. Jaakola, and L. K. Saul. 1999. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*.

K. Kurihara and T. Sato. 2006. Variational Bayesian grammar induction for natural language. In *Proc. of ICGI*.

P. Liang, S. Petrov, M. Jordan, and D. Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proc. of EMNLP*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330.

D. A. Smith and N. A. Smith. 2004. Bilingual parsing with factored estimation: Using English to parse Korean. In *Proc. of EMNLP*, pages 49–56.

N. A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Johns Hopkins University.

B. Snyder and R. Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proc. of ACL*.

Y. W. Teh. 2006. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. of COLING-ACL*.

M. Wang, N. A. Smith, and T. Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for question answering. In *Proc. of EMNLP*.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Comp. Ling.*, 23(3):377–404.

N. Xue, F. Xia, F.-D. Chiou, and M. Palmer. 2004. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 10(4):1–30.

Adding More Languages Improves Unsupervised Multilingual Part-of-Speech Tagging: A Bayesian Non-Parametric Approach

Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and **Regina Barzilay**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{bsnyder, tahira, jacobe, regina}@csail.mit.edu

Abstract

We investigate the problem of unsupervised part-of-speech tagging when raw parallel data is available in a large number of languages. Patterns of ambiguity vary greatly across languages and therefore even unannotated multilingual data can serve as a learning signal. We propose a non-parametric Bayesian model that connects related tagging decisions across languages through the use of multilingual latent variables. Our experiments show that performance improves steadily as the number of languages increases.

1 Introduction

In this paper we investigate the problem of unsupervised part-of-speech tagging when unannotated parallel data is available in a large number of languages. Our goal is to develop a fully joint multilingual model that scales well and shows improved performance for individual languages as the total *number* of languages increases.

Languages exhibit ambiguity at multiple levels, making unsupervised induction of their underlying structure a difficult task. However, sources of linguistic ambiguity vary across languages. For example, the word *fish* in English can be used as either a verb or a noun. In French, however, the noun *poisson* (fish) is entirely distinct from the verbal form *pêcher* (to fish). Previous work has leveraged this idea by building models for unsupervised learning from aligned bilingual data (Snyder et al., 2008). However, aligned data is often available for *many* languages. The benefits of bilingual learning vary markedly depending on which pair of languages is selected, and without labeled data it is unclear how to determine which supplementary language is most helpful. In this paper, we show that it is possible to leverage all aligned languages simultaneously, achieving accuracy that in most cases outperforms even optimally chosen bilingual pairings.

Even in expressing the same meaning, languages take different syntactic routes, leading to variation in part-of-speech sequences. Therefore, an effective multilingual model must accurately model common linguistic structure, yet remain flexible to the idiosyncrasies of each language. This tension only becomes stronger as additional languages are added to the mix. From a computational standpoint, the main challenge is to ensure that the model scales well as the number of languages increases. Care must be taken to avoid an exponential increase in the parameter space as well as the time complexity of inference procedure.

We propose a non-parametric Bayesian model for joint multilingual tagging. The topology of our model connects tagging decisions within a language as well as across languages. The model scales linearly with the number of languages, allowing us to incorporate as many as are available. For each language, the model contains an HMM-like substructure and connects these substructures to one another by means of cross-lingual latent variables. These variables, which we refer to as *superlingual tags*, capture repeated multilingual patterns and thus reduce the overall uncertainty in tagging decisions.

We evaluate our model on a parallel corpus of eight languages. The model is trained once using all

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 83–91,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

languages, and its performance is tested separately for each on a held-out monolingual test set. When a complete tag lexicon is provided, our unsupervised model achieves an average accuracy of 95%, in comparison to 91% for an unsupervised monolingual Bayesian HMM and 97.4% for its supervised counterpart. Thus, on average, the gap between unsupervised and supervised monolingual performance is cut by nearly two thirds. We also examined scenarios where the tag lexicon is reduced in size. In all cases, the multilingual model yielded substantial performance gains. Finally, we examined the performance of our model when trained on all possible subsets of the eight languages. We found that performance improves steadily as the number of available languages increases.

2 Related Work

Bilingual Part-of-Speech Tagging Early work on multilingual tagging focused on projecting annotations from an annotated source language to a target language (Yarowsky and Ngai, 2001; Feldman et al., 2006). In contrast, we assume no labeled data at all; our unsupervised model instead symmetrically improves performance for all languages by learning cross-lingual patterns in raw parallel data. An additional distinction is that projection-based work utilizes pairs of languages, while our approach allows for continuous improvement as languages are added to the mix.

In recent work, Snyder et al. (2008) presented a model for unsupervised part-of-speech tagging trained from a bilingual parallel corpus. This bilingual model and the model presented here share a number of similarities: both are Bayesian graphical models building upon hidden Markov models. However, the bilingual model explicitly joins each aligned word-pair into a single coupled state. Thus, the state-space of these joined nodes grows exponentially in the number of languages. In addition, crossing alignments must be removed so that the resulting graph structure remains acyclic. In contrast, our multilingual model posits latent cross-lingual tags without explicitly joining or directly connecting the part-of-speech tags across languages. Besides permitting crossing alignments, this structure allows the model to scale gracefully with the number of languages.

Beyond Bilingual Learning While most work on multilingual learning focuses on bilingual analysis, some models operate on more than one pair of languages. For instance, Genzel (2005) describes a method for inducing a multilingual lexicon from a group of related languages. His model first induces bilingual models for each pair of languages and then combines them. Our work takes a different approach by simultaneously learning from all languages, rather than combining bilingual results.

A related thread of research is multi-source machine translation (Och and Ney, 2001; Utiyama and Isahara, 2006; Cohn and Lapata, 2007) where the goal is to translate from multiple source languages to a single target language. Rather than jointly training all the languages together, these models train bilingual models separately, and then use their output to select a final translation. The selection criterion can be learned at training time since these models have access to the correct translation. In unsupervised settings, however, we do not have a principled means for selecting among outputs of different bilingual models. By developing a joint multilingual model we can automatically achieve performance that rivals that of the best bilingual pairings.

3 Model

We propose a non-parametric directed Bayesian graphical model for multilingual part-of-speech tagging using a parallel corpus. We perform a joint training pass over the corpus, and then apply the parameters learned for each language to a held-out monolingual test set.

The core idea of our model is that patterns of ambiguity vary across languages and therefore even unannotated multilingual data can serve as a learning signal. Our model is able to simultaneously harness this signal from *all* languages present in the corpus. This goal is achieved by designing a single graphical model that connects tagging decisions within a language as well as across languages.

The model contains language-specific HMM substructures connected to one another by cross-lingual latent variables spanning two or more languages. These variables, which we refer to as *superlingual tags*, capture repeated cross-lingual patterns and

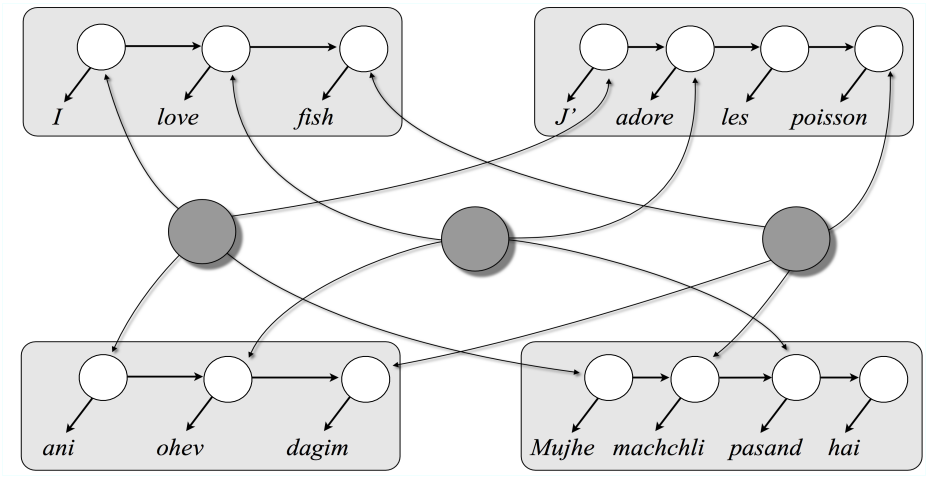


Figure 1: Model structure for parallel sentences in English, French, Hebrew, and Urdu. In this example, there are three superlingual tags, each connected to the part-of-speech tag of a word in each of the four languages.

thus reduce the overall uncertainty in tagging decisions. To encourage the discovery of a compact set of such cross-lingual patterns, we place a Dirichlet process prior on the superlingual tag values.

3.1 Model Structure

For each language, our model includes an HMM-like substructure with observed word nodes, hidden part-of-speech nodes, and directed transition and emission edges. For each set of aligned words in parallel sentences, we add a latent superlingual variable to capture the cross-lingual context. A set of directed edges connect this variable to the part-of-speech nodes of the aligned words. Our model assumes that the superlingual tags for parallel sentences are unordered and are drawn independently of one another.

Edges radiate outward from superlingual tags to language-specific part-of-speech nodes. Thus, our model implicitly assumes that superlingual tags are drawn prior to the part-of-speech tags of all languages and probabilistically influence their selection. See Figure 1 for an example structure.

The particular model structure for each set of parallel sentences (i.e. the configuration of superlingual tags and their edges) is determined by bilingual lexical alignments and — like the text itself — is considered an observed variable. In practice, these lexical alignments are obtained using standard techniques from machine translation.

Our model design has several benefits. Crossing and many-to-many alignments may be used without creating cycles in the graph, as all cross-lingual information emanates from the hidden superlingual tags. Furthermore, the model scales gracefully with the number of languages, as the number of new edges and nodes will be proportional to the number of words for each additional language.

3.2 Superlingual Tags

Each superlingual tag value specifies a set of distributions — one for each language's part-of-speech tagset. In order to learn repeated cross-lingual patterns, we need to constrain the number of superlingual tag values and thus the number of distributions they provide. For example, we might allow the superlingual tags to take on integer values from 1 to K, with each integer value indexing a separate set of distributions. Each set of distributions should correspond to a discovered cross-lingual pattern in the data. For example, one set of distributions might favor nouns in each language and another might favor verbs.

Rather than fixing the number of superlingual tag values to an arbitrary and predetermined size $1, \ldots, K$, we allow them to range over the entire set of integers. In order to encourage the desired multilingual clustering behavior, we use a Dirichlet process prior for the superlingual tags. This prior allows high posterior probability only when a small number

of values are used repeatedly. The actual number of sampled values will be dictated by the data and the number of languages.

More formally, suppose we have n languages, $\ell_1, \ldots, \ell_n$. According to our generative model, a countably infinite sequence of sets $\langle \omega_1^{\ell_1}, \ldots, \omega_1^{\ell_n} \rangle, \langle \omega_2^{\ell_1}, \ldots, \omega_2^{\ell_n} \rangle, \ldots$ is drawn from some base distribution. Each ω_i^ℓ is a distribution over the parts-of-speech in language ℓ.

In parallel, an infinite sequence of mixing components $\pi_1, \pi_2, \ldots$ is drawn from a stick-breaking process (Sethuraman, 1994). These components define a distribution over the integers with most probability mass placed on some initial set of values. The two sequences $\langle \omega_1^{\ell_1}, \ldots, \omega_1^{\ell_n} \rangle, \langle \omega_2^{\ell_1}, \ldots, \omega_2^{\ell_n} \rangle \ldots$ and $\pi_1, \pi_2 \ldots$ now define the distribution over superlingual tags and their associated distributions on parts-of-speech. That is, each superlingual tag $z \in \mathbb{N}$ is drawn with probability π_z, and indexes the set of distributions $\langle \omega_z^{\ell_1}, \ldots, \omega_z^{\ell_n} \rangle$.

3.3 Part-of-Speech Tags

Finally, we need to define the generative probabilities of the part-of-speech nodes. For each such node there may be multiple incoming edges. There will always be an incoming transition edge from the previous tag (in the same language). In addition, there may be incoming edges from zero or more superlingual tags. Each edge carries with it a distribution over parts-of-speech and these distributions must be combined into the single distribution from which the tag is ultimately drawn.

We choose to combine these distributions as a product of experts. More formally: for language ℓ and tag position i, the part-of-speech tag y_i is drawn according to

$$y_i \sim \frac{\phi_{y_{i-1}}(y_i) \prod_z \omega_z^\ell(y_i)}{Z} \tag{1}$$

Where $\phi_{y_{i-1}}$ indicates the transition distribution, and the z's range over the values of the incoming superlingual tags. The normalization term Z is obtained by summing the numerator over all part-of-speech tags y_i in the tagset.

This parameterization allows for a relatively simple and small parameter space. It also leads to a desirable property: for a tag to have high probability *each* of the incoming distributions must allow it. That is, any expert can "veto" a potential tag by assigning it low probability, generally leading to consensus decisions.

We now formalize this description by giving the stochastic process by which the observed data (raw parallel text) is generated, according to our model.

3.4 Generative Process

For n languages, we assume the existence of n tagsets $T^1, \ldots, T^n$ and vocabularies, $W^1, \ldots, W^n$, one for each language. For clarity, the generative process is described using only bigram transition dependencies, but our experiments use a trigram model.

1. **Transition and Emission Parameters:** For each language ℓ and for each tag $t \in T^\ell$, draw a *transition* distribution ϕ_t^ℓ over tags T_ℓ and an *emission* distribution θ_t^ℓ over words W^ℓ, all from symmetric Dirichlet priors of appropriate dimension.

2. **Superlingual Tag Parameters:** Draw an infinite sequence of sets $\langle \omega_1^{\ell_1}, \ldots, \omega_1^{\ell_n} \rangle, \langle \omega_2^{\ell_1}, \ldots, \omega_2^{\ell_n} \rangle, \ldots$ from base distribution G_0. Each ω_i^ℓ is a distribution over the tagset T^ℓ. The base distribution G_0 is a product of n symmetric Dirichlets, where the dimension of the i^{th} such Dirichlet is the size of the corresponding tagset T^{ℓ_i}.

   At the same time, draw an infinite sequence of mixture weights $\pi \sim GEM(\alpha)$, where $GEM(\alpha)$ indicates the stick-breaking distribution (Sethuraman, 1994), and $\alpha = 1$. These parameters together define a prior distribution over superlingual tags,

$$p(z) = \sum_k^\infty \pi_k \delta_{k=z}, \tag{2}$$

   or equivalently over the part-of-speech distributions $\langle \omega^{\ell_1}, \ldots, \omega^{\ell_n} \rangle$ that they index:

$$\sum_k^\infty \pi_k \delta_{\langle \omega_k^{\ell_1}, \ldots, \omega_k^{\ell_n} \rangle = \langle \omega^{\ell_1}, \ldots, \omega^{\ell_n} \rangle}. \tag{3}$$

   In both cases, $\delta_{v=v'}$ is defined as one when $v = v'$ and zero otherwise. Distribution 3 is said to be drawn from a Dirichlet process, conventionally written as $DP(\alpha, G_0)$.

3. **Data:** For each multilingual parallel sentence,

   (a) Draw an alignment a specifying sets of aligned indices across languages. Each such set may consist of indices in any subset of the languages. We leave the distribution over alignments undefined, as we consider alignments observed variables.

   (b) For each set of indices in a, draw a superlingual tag value z according to Distribution 2.

   (c) For each language ℓ, for $i = 1, \ldots$ (until end-tag reached):

      i. Draw a part-of-speech tag $y_i \in T^\ell$ according to Distribution 1

      ii. Draw a word $w_i \in W^\ell$ according to the emission distribution θ_{y_i}.

To perform Bayesian inference under this model we use a combination of sampling techniques, which we describe in detail in the next section.

3.5 Inference

Ideally we would like to predict the part-of-speech tags which have highest *marginal* probability given the observed words $\mathbf{x}$ and alignments $\mathbf{a}$. More specifically, since we are evaluating our accuracy per tag-position, we would like to predict, for language index ℓ and word index i, the single part-of-speech tag:

$$\operatorname*{argmax}_{t \in T^\ell} P\big(y_i^\ell = t \big| \mathbf{x}, \mathbf{a}\big)$$

which we can rewrite as the $\operatorname{argmax}_{t \in T^\ell}$ of the integral,

$$\int \Bigg[P\left(y_i^\ell = t \Big| \mathbf{y}_{-(\ell,i)}, \boldsymbol{\phi}, \boldsymbol{\theta}, \mathbf{z}, \boldsymbol{\omega}, \mathbf{x}, \mathbf{a}\right) \cdot$$
$$P\left(\mathbf{y}_{-(\ell,i)}, \boldsymbol{\phi}, \boldsymbol{\theta}, \mathbf{z}, \boldsymbol{\pi}, \boldsymbol{\omega}, \Big| \mathbf{x}, \mathbf{a}\right) \Bigg] d\mathbf{y}_{-(\ell,i)}\, d\boldsymbol{\phi}\, d\boldsymbol{\theta}\, d\mathbf{z}\, d\boldsymbol{\pi}\, d\boldsymbol{\omega},$$

in which we marginalize over the settings of all tags other than y_i^ℓ (written as $\mathbf{y}_{-(\ell,i)}$), the transition distributions $\boldsymbol{\phi} = \left\{\phi_{t'}^\ell\right\}$, emission distributions $\boldsymbol{\theta} = \left\{\theta_{t'}^\ell\right\}$, superlingual tags $\mathbf{z}$, and superlingual tag parameters $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots\}$ and $\boldsymbol{\omega} = \left\{\langle \omega_1^{\ell_1}, \ldots, \omega_1^{\ell_n} \rangle, \langle \omega_2^{\ell_1}, \ldots, \omega_2^{\ell_n} \rangle \ldots\right\}$ (where t' ranges over all part-of-speech tags).

As these integrals are intractable to compute exactly, we resort to the standard Monte Carlo approximation. We collect N samples of the variables over which we wish to marginalize but for which we cannot compute closed-form integrals, where each sample $sample_k$ is drawn from $P(sample_k|\mathbf{x}, \mathbf{a})$. We then approximate the tag marginals as:

$$P\big(y_i^\ell = t \big| \mathbf{x}, \mathbf{a}\big) \approx \frac{\sum_k P\big(y_i^\ell = t \big| sample_k, \mathbf{x}, \mathbf{a}\big)}{N} \quad (4)$$

We employ closed forms for integrating out the emission parameters $\boldsymbol{\theta}$, transition parameters $\boldsymbol{\phi}$, and superlingual tag parameters $\boldsymbol{\pi}$ and $\boldsymbol{\omega}$. We explicitly sample only part-of-speech tags $\mathbf{y}$, superlingual tags $\mathbf{z}$, and the hyperparameters of the transition and emission Dirichlet priors. To do so, we apply standard Markov chain sampling techniques: a Gibbs sampler for the tags and a within-Gibbs Metropolis-Hastings subroutine for the hyperparameters (Hastings, 1970).

Our Gibbs sampler samples each part-of-speech and superlingual tag separately, conditioned on the current value of all other tags. In each case, we use standard closed forms to integrate over all parameter values, using currently sampled counts and hyperparameter pseudo-counts. We note that conjugacy is technically broken by our use of a product form in Distribution 1. Nevertheless, we consider the sampled tags to have been generated separately by each of the factors involved in the numerator. Thus our method of using count-based closed forms should be viewed as an approximation.

3.6 Sampling Part-of-Speech Tags

To sample the part-of-speech tag for language ℓ at position i we draw from

$$P(y_i^\ell | \mathbf{y}_{-(\ell,i)}, \mathbf{x}, \mathbf{a}, \mathbf{z}) \propto$$
$$P(y_{i+1}^\ell | y_i^\ell, \mathbf{y}_{-(\ell,i)}, \mathbf{a}, \mathbf{z})\; P(y_i^\ell | \mathbf{y}_{-(\ell,i)}, \mathbf{a}, \mathbf{z}) \cdot$$
$$P(x_i^\ell | \mathbf{x}_{-i}^\ell, \mathbf{y}^\ell)\,,$$

where the first two terms are the generative probabilities of *(i)* the current tag given the previous tag and superlingual tags, and *(ii)* the next tag given the current tag and superlingual tags. These two quantities are similar to Distribution 1, except here we integrate over the transition parameter $\phi_{y_{i-1}}$ and the superlingual tag parameters ω_z^ℓ. We end up with a product of integrals. Each integral can be computed in closed form using multinomial-Dirichlet conjugacy (and by making the above-mentioned simplifying assumption that all other tags were generated separately by their transition and superlingual

parameters), just as in the monolingual Bayesian HMM of (Goldwater and Griffiths, 2007).

For example, the closed form for integrating over the parameter of a superlingual tag with value z is given by:

$$\int \omega_z^\ell(y_i) P(\omega_z^\ell|\omega_0) d\omega_z^\ell = \frac{count(z, y_i, \ell) + \omega_0}{count(z, \ell) + T^\ell \omega_0}$$

where $count(z, y_i, \ell)$ is the number of times that tag y_i is observed together with superlingual tag z in language ℓ, $count(z, \ell)$ is the total number of times that superlingual tag z appears with an edge into language ℓ, and ω_0 is a hyperparameter.

The third term in the sampling formula is the emission probability of the current word x_i^ℓ given the current tag and all other words and sampled tags, as well as a hyperparameter which is suppressed for the sake of clarity. This quantity can be computed exactly in closed form in a similar way.

3.7 Sampling Superlingual Tags

For each set of aligned words in the observed alignment $\mathbf{a}$ we need to sample a superlingual tag z. Recall that z is an index into an infinite sequence $\langle \omega_1^{\ell_1}, \ldots, \omega_1^{\ell_n} \rangle, \langle \omega_2^{\ell_1}, \ldots, \omega_2^{\ell_n} \rangle \ldots$, where each ω_z^ℓ is a distribution over the tagset T^ℓ. The generative distribution over z is given by equation 2. In our sampling scheme, however, we integrate over all possible settings of the mixing components $\boldsymbol{\pi}$ using the standard Chinese Restaurant Process (CRP) closed form (Antoniak, 1974):

$$P(z_i|\mathbf{z}_{-i}, \mathbf{y}) \propto$$
$$\prod_\ell P(y_i^\ell|\mathbf{z}, \mathbf{y}_{-(\ell,i)}) \cdot \begin{cases} \frac{1}{k+\alpha} count(z_i) & \text{if } z_i \in \mathbf{z}_{-i} \\ \frac{\alpha}{k+\alpha} & \text{otherwise} \end{cases}$$

The first term is the product of closed form tag probabilities of the aligned words, given z. The final term is the standard CRP closed form for posterior sampling from a Dirichlet process prior. In this term, k is the total number of sampled superlingual tags, $count(z_i)$ is the total number of times the value z_i occurs in the sampled tags, and α is the Dirichlet process concentration parameter (see Step 2 in Section 3.4).

Finally, we perform standard hyperparameter reestimation for the parameters of the Dirichlet distribution priors on $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ (the transition and emission distributions) using Metropolis-Hastings. We assume an improper uniform prior and use a Gaussian proposal distribution with mean set to the previous value, and variance to one-tenth of the mean.

4 Experimental Setup

We test our model in an unsupervised framework where only raw parallel text is available for each of the languages. In addition, we assume that for each language a tag dictionary is available that covers some subset of words in the text. The task is to learn an independent tagger for each language that can annotate non-parallel raw text using the learned parameters. All reported results are on non-parallel monolingual test data.

Data For our experiments we use the Multext-East parallel corpus (Erjavec, 2004) which has been used before for multilingual learning (Feldman et al., 2006; Snyder et al., 2008). The tagged portion of the corpus includes a 100,000 word English text, Orwell's novel "Nineteen Eighty Four", and its translation into seven languages: Bulgarian, Czech, Estonian, Hungarian, Romanian, Slovene and Serbian. The corpus also includes a tag lexicon for each of these languages. We use the first 3/4 of the text for learning and the last 1/4 as held-out non-parallel test data.

The corpus provides sentence level alignments. To obtain word level alignments, we run GIZA++ (Och and Ney, 2003) on all 28 pairings of the 8 languages. Since we want each latent superlingual variable to span as many languages as possible, we aggregate the pairwise lexical alignments into larger sets of aligned words. These sets of aligned words are generated as a preprocessing step. During sampling they remain fixed and are treated as observed data.

We use the set of 14 basic part-of-speech tags provided by the corpus. In our first experiment, we assume that a complete tag lexicon is available, so that for each word, its set of possible parts-of-speech is known ahead of time. In this setting, the average number of possible tags per token is 1.39. We also experimented with incomplete tag dictionaries, where entries are only available for words appearing more than five or ten times in the corpus. For other words, the entire tagset of 14 tags is considered. In these two scenarios, the average per-token tag ambi-

	Lexicon: Full				Lexicon: Frequency > 5				Lexicon: Frequency > 10			
	MONO	BI		MULTI	MONO	BI		MULTI	MONO	BI		MULTI
		AVG	BEST			AVG	BEST			AVG	BEST	
BG	88.8	91.3	**94.7**	92.6	73.5	80.2	**82.7**	81.3	71.9	77.8	**80.2**	78.8
CS	93.7	97.0	97.7	**98.2**	72.2	79.0	79.7	**83.0**	66.7	75.3	76.7	**79.4**
EN	95.8	95.9	**96.1**	95.0	87.3	90.4	**90.7**	88.1	84.4	88.8	**89.4**	86.1
ET	92.5	93.4	94.3	**94.6**	72.5	76.5	77.5	**80.6**	68.3	72.9	74.9	**77.9**
HU	95.3	96.8	**96.9**	96.7	73.5	77.3	78.0	**80.8**	69.0	73.8	75.2	**76.4**
RO	90.1	91.8	94.0	**95.1**	77.1	82.7	84.4	**86.1**	73.0	80.5	82.1	**83.1**
SL	87.4	89.3	94.8	**95.8**	75.7	78.7	80.9	**83.6**	70.4	76.1	77.6	**80.0**
SR	84.5	90.2	**94.5**	92.3	66.3	75.9	**79.4**	78.8	63.7	72.4	**76.1**	75.9
Avg.	91.0	93.2	**95.4**	95.0	74.7	80.1	81.7	**82.8**	70.9	77.2	79.0	**79.7**

Table 1: Tagging accuracy for Bulgarian, Czech, English, Estonian, Hungarian, Romanian, Slovene, and Serbian. In the first scenario, a complete tag lexicon is available for all the words. In the other two scenarios the tag lexicon only includes words that appear more than five or ten times. Results are given for a monolingual Bayesian HMM (Goldwater and Griffiths, 2007), a bilingual model (Snyder et al., 2008), and the multilingual model presented here. In the case of the bilingual model, we present both the average accuracy over all pairings as well as the result from the best performing pairing for each language. The best results for each language in each scenario are given in boldface.

guity is 4.65 and 5.58, respectively.

Training and testing In the full lexicon experiment, each word is initialized with a random part-of-speech tag from its dictionary entry. In the two reduced lexicon experiments, we initialize the tags with the result of our monolingual baseline (see below) to reduce sampling time. In both cases, we begin with 14 superlingual tag values — corresponding to the parts-of-speech — and initially assign them based on the most common initial part-of-speech of words in each alignment.

We run our Gibbs sampler for 1,000 iterations, and store the conditional tag probabilities for the last 100 iterations. We then approximate marginal tag probabilities on the training data using Equation 4 and predict the highest probability tags. Finally, we compute maximum likelihood transition and emission probabilities using these tag counts, and then apply smoothed viterbi decoding to each held-out monolingual test set. All reported results are averaged over five runs of the sampler.

Monolingual and bilingual baselines We reimplemented the Bayesian HMM model of Goldwater and Griffiths (2007) (BHMM1) as our monolingual baseline. It has a standard HMM structure with conjugate Bayesian priors over transitions and emissions. We note that our model, in the absence of any superlingual tags, reduces to this Bayesian HMM. As an additional baseline we use a bilingual model (Snyder et al., 2008). It is a directed graphical model that jointly tags two parallel streams of text aligned at the word level. The structure of the model consists of two parallel HMMs, one for each language. The aligned words form joint nodes that are shared by both HMMs. These joint nodes are sampled from a probability distribution that is a product of the transition and emission distributions in the two languages and a coupling distribution.

We note that the numbers reported here for the bilingual model differ slightly from those reported by Snyder et al. (2008) for two reasons: we use a slightly larger set of sentences, and an improved sampling scheme. The new sampling scheme marginalizes over the transition and coupling parameters by using the same count-based approximation that we utilize for our multilingual model. This leads to higher performance, and thus a stronger baseline.[1]

5 Results

Table 1 shows the tagging accuracy of our multilingual model on the test data, when training is performed on all eight languages together. Results from both baselines are also reported. In the case of the bilingual baseline, seven pairings are possible for each language, and the results vary by pair. There-

[1] Another difference is that we use the English lexicon provided with the Multext-East corpus, whereas (Snyder et al., 2008) augment this lexicon with tags found in WSJ.

fore, for each language, we present the average accuracy over all seven pairings, as well as the accuracy of its highest performing pairing.

We provide results for three scenarios. In the first case, a tag dictionary is provided for all words, limiting them to a restricted set of possible tags. In the other two scenarios, dictionary entries are limited to words that appear more than five or ten times in the corpus. All other words can be assigned any tag, increasing the overall difficulty of the task. In the full lexicon scenario, our model achieves an average tagging accuracy of 95%, compared to 91% for the monolingual baseline and 93.2% for the bilingual baseline when averaged over all pairings. This accuracy (95%) comes close to the performance of the bilingual model when the best pairing for each language is chosen by an oracle (95.4%). This demonstrates that our multilingual model is able to effectively learn from all languages. In the two reduced lexicon scenarios, the gains are even more striking. In both cases the average multilingual performance outpaces even the *best* performing pairs.

Looking at individual languages, we see that in all three scenarios, Czech, Estonian, Romanian, and Slovene show their best performance with the multilingual model. Bulgarian and Serbian, on the other hand, give somewhat better performance with their optimal pairings under the bilingual model, but their multilingual performance remains higher than their average bilingual results. The performance of English under the multilingual model is somewhat lower, especially in the full lexicon scenario, where it drops below monolingual performance. One possible explanation for this decrease lies in the fact that English, by far, has the lowest trigram tag entropy of all eight languages (Snyder et al., 2008). It is possible, therefore, that the signal it should be getting from its own transitions is being drowned out by less reliable information from other languages.

In order to test the performance of our model as the number of languages increases, we ran the full lexicon experiment with all possible subsets of the eight languages. Figure 2 plots the average accuracy as the number of languages varies. For comparison, the monolingual and average bilingual baseline results are given, along with supervised monolingual performance. Our multilingual model steadily gains in accuracy as the number of available languages increases. Interestingly, it even outperforms the bilingual baseline (by a small margin) when only two languages are available, which may be attributable to the more flexible non-parametric dependencies employed here. Finally, notice that the gap between monolingual supervised and unsupervised performance is cut by nearly two thirds under the unsupervised multilingual model.

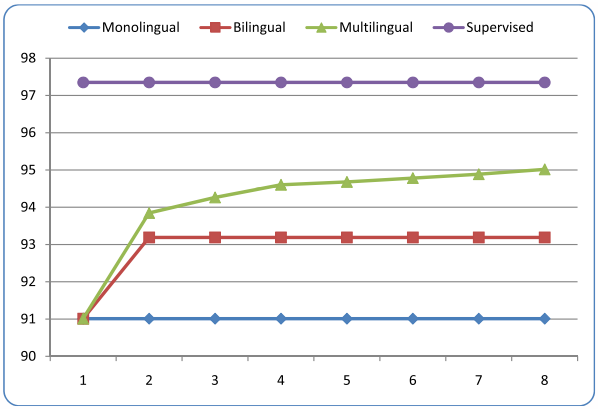


Figure 2: Performance of the multilingual model as the number of languages varies. Performance of the monolingual and average bilingual baselines as well as a supervised monolingual performance are given for comparison.

6 Conclusion

In this paper we've demonstrated that the benefits of unsupervised multilingual learning increase steadily with the number of available languages. Our model scales gracefully as languages are added and effectively incorporates information from them all, leading to substantial performance gains. In one experiment, we cut the gap between unsupervised and supervised performance by nearly two thirds. A future challenge lies in incorporating constraints from additional languages even when parallel text is unavailable.

Acknowledgments

The authors acknowledge the support of the National Science Foundation (CAREER grant IIS-0448168 and grant IIS-0835445). Thanks to Tommi Jaakkola and members of the MIT NLP group for helpful discussions. Any opinions, findings, or recommendations expressed above are those of the authors and do not necessarily reflect the views of the NSF.

References

C. E. Antoniak. 1974. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2:1152–1174, November.

Trevor Cohn and Mirella Lapata. 2007. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proceedings of ACL*.

T. Erjavec. 2004. MULTEXT-East version 3: Multilingual morphosyntactic specifications, lexicons and corpora. In *Fourth International Conference on Language Resources and Evaluation, LREC*, volume 4, pages 1535–1538.

Anna Feldman, Jirka Hana, and Chris Brew. 2006. A cross-language approach to rapid creation of new morpho-syntactically annotated resources. In *Proceedings of LREC*, pages 549–554.

Dmitriy Genzel. 2005. Inducing a multilingual dictionary from a parallel multitext in related languages. In *Proceedings of the HLT/EMNLP*, pages 875–882.

Sharon Goldwater and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the ACL*, pages 744–751.

W. K. Hastings. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109.

Franz Josef Och and Hermann Ney. 2001. Statistical multi-source translation. In *MT Summit 2001*, pages 253–258.

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

J. Sethuraman. 1994. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650.

Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. 2008. Unsupervised multilingual learning for POS tagging. In *Proceedings of the EMNLP*, pages 1041–1050.

Masao Utiyama and Hitoshi Isahara. 2006. A comparison of pivot methods for phrase-based statistical machine translation. In *Proceedings of NAACL/HLT*, pages 484–491.

David Yarowsky and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Proceedings of the NAACL*, pages 1–8.

Efficiently Parsable Extensions to Tree-Local Multicomponent TAG

Rebecca Nesson
School of Engineering
and Applied Sciences
Harvard University
Cambridge, MA
nesson@seas.harvard.edu

Stuart M. Shieber
School of Engineering
and Applied Sciences
Harvard University
Cambridge, MA
shieber@seas.harvard.edu

Abstract

Recent applications of Tree-Adjoining Grammar (TAG) to the domain of semantics as well as new attention to syntactic phenomena have given rise to increased interested in more expressive and complex multicomponent TAG formalisms (MCTAG). Although many constructions can be modeled using tree-local MCTAG (TL-MCTAG), certain applications require even more flexibility. In this paper we suggest a shift in focus from constraining locality and complexity through tree- and set-locality to constraining locality and complexity through restrictions on the derivational distance between trees in the same tree set in a valid derivation. We examine three formalisms, restricted NS-MCTAG, restricted Vector-TAG and delayed TL-MCTAG, that use notions of derivational distance to constrain locality and demonstrate how they permit additional expressivity beyond TL-MCTAG without increasing complexity to the level of set local MCTAG.

1 Introduction

Tree-Adjoining Grammar (TAG) has long been popular for natural language applications because of its ability to naturally capture syntactic relationships while also remaining efficient to process. More recent applications of TAG to the domain of semantics as well as new attention to syntactic phenomena such as scrambling have given rise to increased interested in multicomponent TAG formalisms (MCTAG), which extend the flexibility, and in some cases generative capacity of the formalism but also have substantial costs in terms of efficient processing. Much work in TAG semantics makes use of tree-local MCTAG (TL-MCTAG) to model phenomena such as quantifier scoping, Wh-question formation, and many other constructions (Kallmeyer and Romero, 2004; Romero et al., 2004). Certain applications, however, appear to require even more flexibility than is provided by TL-MCTAG. Scrambling is one well-known example (Rambow, 1994). In addition, in the semantics domain, the use of a new TAG operation, flexible composition, is used to perform certain semantic operations that seemingly cannot be modeled with TL-MCTAG alone (Chiang and Scheffler, 2008) and in work in synchronous TAG semantics, constructions such as nested quantifiers require a set-local MCTAG (SL-MCTAG) analysis (Nesson and Shieber, 2006).

In this paper we suggest a shift in focus from constraining locality and complexity through restrictions that all trees in a tree set must adjoin within a single tree or tree set to constraining locality and complexity through restrictions on the derivational distance between trees in the same tree set in a valid derivation. We examine three formalisms, two of them introduced in this work for the first time, that use derivational distance to constrain locality and demonstrate by construction of parsers their relationship to TL-MCTAG in both expressivity and complexity. In Section 2 we give a very brief introduction to TAG. In Section 3 we elaborate further the distinction between these two types of locality restrictions using TAG derivation trees. Section 4 briefly addresses the simultaneity requirement present in MCTAG formalisms but not in Vector-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 92–100,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

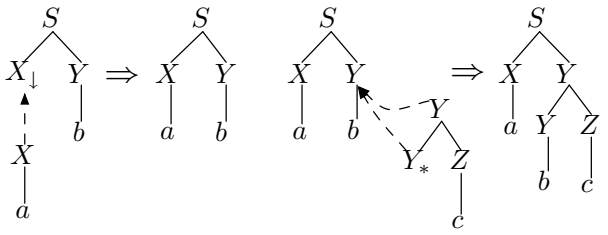


Figure 1: An example of the TAG operations **substitution** and **adjunction**.

TAG formalisms and argues for dropping the requirement. In Sections 5 and 6 we introduce two novel formalisms, restricted non-simultaneous MC-TAG and restricted Vector-TAG, respectively, and define CKY-style parsers for them. In Section 7 we recall the delayed TL-MCTAG formalism introduced by Chiang and Scheffler (2008) and define a CKY-style parser for it as well. In Section 8 we explore the complexity of all three parsers and the relationship between the formalisms. In Section 9 we discuss the linguistic applications of these formalisms and show that they permit analyses of some of the hard cases that have led researchers to look beyond TL-MCTAG.

2 Background

A tree-adjoining grammar consists of a set of elementary tree structures of arbitrary depth, which are combined by operations of **adjunction** and **substitution**. **Auxiliary trees** are elementary trees in which the root and a frontier node, called the **foot node** and distinguished by the diacritic $*$, are labeled with the same nonterminal A. The adjunction operation entails splicing an auxiliary tree in at an internal node in an elementary tree also labeled with nonterminal A. Trees without a foot node, which serve as a base for derivations and may combine with other trees by substitution, are called **initial trees**. Examples of the adjunction and substitution operations are given in Figure 1. For further background, refer to the survey by (Joshi and Schabes, 1997).

Shieber et al. (1995) and Vijay-Shanker (1987) apply the Cocke-Kasami-Younger (CKY) algorithm first introduced for use with context-free grammars in Chomsky normal form (Kasami, 1965; Younger, 1967) to the TAG parsing problem to generate parsers with a time complexity of $O(n^6|G|^2)$. In order to clarify the presentation of our extended TL-MCTAG parsers below, we briefly review the algorithm of Shieber et al. (1995) using the inference rule notation from that paper. As shown in Figure 2, items in CKY-style TAG parsing consist of a node in an elementary tree and the indices that mark the edges of the span dominated by that node. Nodes, notated $\alpha@a^\circ$, are specified by three pieces of information: the identifier α of the elementary tree the node is in, the Gorn address a of the node in that tree[1], and a diacritic, $\circ$, indicating that an adjunction or substitution is still available at that node or $\bullet$, indicating that one has already taken place.

Each item has four indices, indicating the left and right edges of the span covered by the node as well as any gap in the node that may be the result of a foot node dominated by the node. Nodes that do not dominate a foot node will have no gap in them, which we indicate by the use of underscores in place of the indices for the gap. To limit the number of inference rules needed, we define the following function $i \cup j$ for combining indices:

$$i \cup j = \begin{cases} i & j = \_ \\ j & i = \_ \\ i & i = j \\ \text{undefined} & \text{otherwise} \end{cases}$$

The side conditions $\mathsf{Init}(\alpha)$ and $\mathsf{Aux}(\alpha)$ hold if α is an initial tree or an auxiliary tree, respectively. $\mathsf{Label}(\alpha@a)$ specifies the label of the node in tree α at address a. $\mathsf{Ft}(\alpha)$ specifies the address of the foot node of tree α. $\mathsf{Adj}(\alpha@a, \beta)$ holds if tree β may adjoin into tree α at address a. $\mathsf{Subst}(\alpha@a, \beta)$ holds if tree β may substitute into tree α at address a. These conditions fail if the adjunction or substitution is prevented by constraints such as mismatched node labels.

Multi-component TAG (MCTAG) generalizes TAG by allowing the elementary items to be sets of trees rather than single trees (Joshi and Schabes, 1997). The basic operations are the same but all trees in a set must adjoin (or substitute) into another tree or tree set in a single step in the derivation.

An MCTAG is **tree-local** if tree sets are required to adjoin within a single elementary tree (Weir,

[1] A Gorn address uniquely identifies a node within a tree. The Gorn address of the root node is ε. The jth child of the node with address i has address $i \cdot j$.

Goal Item:	$\langle \alpha@\varepsilon^\bullet, 0, \_, \_, n \rangle$	$\mathsf{Init}(\alpha)$ $\mathsf{Label}(\alpha@\varepsilon) = S$
Terminal Axiom:	$\langle \alpha@a^\bullet, i-1, \_, \_, i \rangle$	$\mathsf{Label}(\alpha@a) = w_i$
Empty Axiom:	$\langle \alpha@a^\bullet, i, \_, \_, i \rangle$	$\mathsf{Label}(\alpha@a) = \varepsilon$
Foot Axiom:	$\langle \alpha@\mathsf{Ft}(\alpha)^\circ, p, p, q, q \rangle$	$\mathsf{Aux}(\alpha)$
Unary Complete:	$\dfrac{\langle \alpha@(a \cdot 1)^\bullet, i, j, k, l \rangle}{\langle \alpha@a^\circ, i, j, k, l \rangle}$	$\alpha@(a \cdot 2)$ undefined
Binary Complete:	$\dfrac{\langle \alpha@(a \cdot 1)^\bullet, i, j, k, l \rangle, \langle \alpha@(a \cdot 2)^\bullet, l, j', k', m \rangle}{\langle \alpha@a^\circ, i, j \cup j', k \cup k', m \rangle}$	
Adjoin:	$\dfrac{\langle \beta@\varepsilon^\bullet, i, p, q, l \rangle, \langle \alpha@a^\circ, p, j, k, q \rangle}{\langle \alpha@a^\bullet, i, j, k, l \rangle}$	$\mathsf{Adj}(\alpha@a, \beta)$
No Adjoin:	$\dfrac{\langle \alpha@a^\circ, i, j, k, l \rangle}{\langle \alpha@a^\bullet, i, j, k, l \rangle}$	
Substitute:	$\dfrac{\langle \beta@\varepsilon^\bullet, i, \_, \_, l \rangle}{\langle \alpha@a^\bullet, i, \_, \_, l \rangle}$	$\mathsf{Subst}(\alpha@a, \beta)$

Figure 2: The CKY algorithm for TAG

1988). Although tree-local MCTAG (TL-MCTAG) has the same generative capacity as TAG (Weir, 1988), the conversion to TAG is exponential and the TL-MCTAG formalism is NP-hard to recognize (Søgaard et al., 2007). An MCTAG is **set-local** if tree sets required to adjoin within a single elementary tree set (Weir, 1988). Set-local MCTAG (SL-MCTAG) has equivalent expressivity to linear context-free rewriting systems and recognition is provably PSPACE complete (Nesson et al., 2008).

3 Domains of Locality and Derivation Trees

The domains of locality of TL-MCTAG and SL-MCTAG (and trivially, TAG) can be thought of as lexically defined. That is, all locations at which the adjunction of one tree set into another may occur must be present within a single lexical item. However, we can also think of locality derivationally. In a derivationally local system the constraint is on the relationships allowed between members of the same tree set in the derivation tree.

TAG derivation trees provide the information about how the elementary structures of the grammar combine that is necessary to construct the derived tree. Nodes in a TAG derivation tree are labeled with identifiers of elementary structures. One elementary structure is the child of another in the derivation tree if it adjoins or substitutes into it in the derivation. Arcs in the derivation tree are labeled with the address in the target elementary structure at which the operation takes place.

In MCTAG the derivation trees are often drawn with identifiers of entire tree sets as the nodes of the tree because the lexical locality constraints require that each elementary tree set be the derivational child of only one other tree set. However, if we elaborate the derivation tree to include a node for each tree in the grammar rather than only for each tree set we can see a stark contrast in the derivational

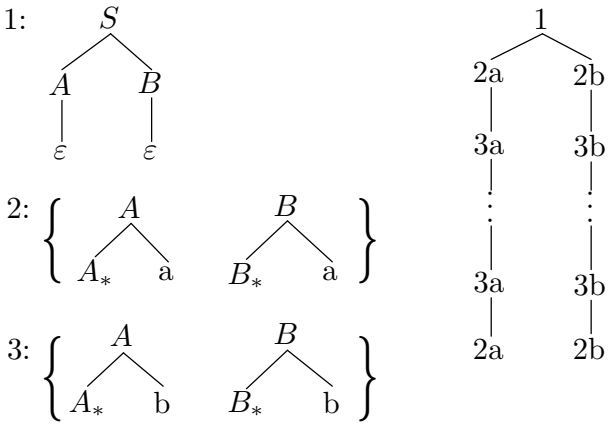


Figure 3: An example SL-MCTAG grammar that generates the language *ww* and associated derivation tree that demonstrating an arbitrarily long derivational distance between the trees of a given tree set and their nearest common ancestor. Note that if this grammar is interpreted as a TL-MCTAG grammar only two derivations are possible (for the strings *aa* and *bb*).

locality of these two formalisms. In TL-MCTAG all trees in a set must adjoin to the same tree. This means that they must all be siblings in the derivation tree. In SL-MCTAG, on the other hand, it is possible to generate derivations with arbitrarily long distances before the nearest common ancestor of two trees from the same elementary tree set is reached. An example SL-MCTAG grammar that can produce an arbitrarily long derivational distance to the nearest common ancestor of the trees in a given tree set is given in Figure 3.

Chiang and Scheffler (2008) recently introduced one variant of MCTAG, delayed Tree-Local MCTAG (delayed TL-MCTAG) that uses a derivational notion of locality. In this paper we introduce two additional derivationally local TAG-based formalisms, restricted non-simultaneous MCTAG (restricted NS-MCTAG) and restricted Vector TAG (restricted V-TAG) and demonstrate by construction of parsers how each gives rise to a hierarchy of derivationally local formalisms with a well-defined efficiency penalty for each step of derivational distance permitted.

4 The Simultaneity Requirement

In addition to lexical locality constraints the definition of MCTAG requires that all trees from a set adjoin simultaneously. In terms of well-formed derivation trees, this amounts to disallowing derivations in which a tree from a given set is the ancestor of a tree from the same tree set. For most linguistic applications of TAG, this requirement seems natural and is strictly obeyed. There are a few applications, including flexible composition and scrambling in free-word order languages that benefit from TAG-based grammars that drop the simultaneity requirement (Chiang and Scheffler, 2008; Rambow, 1994). From a complexity perspective, however, checking the simultaneity requirement is expensive (Kallmeyer, 2007). As a result, it can be advantageous to select a base formalism that does not require simultaneity even if the grammars implemented with it do not make use of that additional freedom.

5 Restricted Non-simultaneous MCTAG

The simplest version of a derivationally local TAG-based formalism is most similar to non-local MCTAG. There is no lexical locality requirement at all. In addition, we drop the simultaneity requirement. Thus the only constraint on elementary tree sets is the limit, d, on the derivational distance between the trees in a given set and their nearest common ancestor. We call this formalism restricted non-simultaneous MCTAG. Note that if we constrain d to be one, this happens to enforce both the derivational delay limit and the lexical locality requirement of TL-MCTAG.

A CKY-style parser for restricted NS-MCTAG with a restriction of d is given in Figure 4. The items of this parser contain d lists, $\Lambda^1, \ldots, \Lambda^d$, called **histories** that record the identities of the trees that have already adjoined in the derivation in order to enforce the locality constraints. The identities of the trees in a tree set that have adjoined in a given derivation are maintained in the histories until all the trees from that set have adjoined. Once the locality constraint is checked for a tree set, the Filter side condition expunges those trees from the histories. A tree is recorded in this history list with superscript i, where i is the derivational distance between the location where the recorded tree adjoined and the location of the current item. The locality constraint is enforced at the point of adjunction or substitution where the

Goal Item

$$\langle \alpha_0@\varepsilon^\bullet, 0, \_, \_, n, \emptyset, \ldots, \emptyset \rangle$$

$\mathsf{Init}(\alpha_1)$
$\mathsf{Label}(\alpha_0@\varepsilon) = S$
$|\alpha| = 1$

Terminal Axiom

$$\langle \alpha_x@a^\bullet, i-1, \_, \_, i, \emptyset, \ldots, \emptyset \rangle$$

$\mathsf{Label}(\alpha_x@a) = w_i$

Empty Axiom

$$\langle \alpha_x@a^\bullet, i, \_, \_, i, \emptyset, \ldots, \emptyset \rangle$$

$\mathsf{Label}(\alpha_x@a) = \varepsilon$

Foot Axiom

$$\langle \alpha_x@\mathsf{Ft}(\alpha_x)^\circ, p, p, q, q, \emptyset, \ldots, \emptyset \rangle$$

$\mathsf{Aux}(\alpha_x)$

Unary Complete

$$\frac{\langle \alpha_x@(a \cdot 1)^\bullet, i, j, k, l, \Lambda^1, \ldots, \Lambda^d \rangle}{\langle \alpha_x@a^\circ, i, j, k, l, \Lambda^1, \ldots, \Lambda^d \rangle}$$

$\alpha_x@(a \cdot 2)$ undefined

Binary Complete

$$\frac{\langle \alpha_x@(a \cdot 1)^\bullet, i, j, k, l, \Lambda_1^1, \ldots, \Lambda_1^d \rangle \langle \alpha_x@(a \cdot 2)^\bullet, l, j', k', m, \Lambda_2^1, \ldots, \Lambda_2^d \rangle}{\langle \alpha_x@a^\circ, i, j \cup j', k \cup k', m, \Lambda^1, \ldots, \Lambda^d \rangle}$$

$\mathsf{Filter}(\Lambda_1^1 \cup \Lambda_2^1, \ldots, \Lambda_1^d \cup \Lambda_2^d) = \Lambda^1, \ldots, \Lambda^d$

Adjoin:

$$\frac{\langle \beta_y@\varepsilon^\bullet, i, p, q, l, \Lambda_1^1, \ldots, \Lambda_1^{d-1}, \emptyset \rangle \langle \alpha_x@a^\circ, p, j, k, q, \Lambda_2^1, \ldots, \Lambda_2^d \rangle}{\langle \alpha_x@a^\bullet, i, j, k, l, \Lambda^1, \ldots, \Lambda^d \rangle}$$

$\mathsf{Adj}(\alpha_x@a, \beta_y)$
$\mathsf{Filter}(\Lambda_2^1 \cup \{\beta_y\}, \Lambda_2^2 \cup \Lambda_1^1, \ldots, \Lambda_2^d \cup \Lambda_1^{d-1}) = \Lambda^1, \ldots, \Lambda^d$

Substitute:

$$\frac{\langle \beta_y@\varepsilon^\bullet, i, \_, \_, l, \Lambda_1^1, \ldots, \Lambda_1^{d-1}, \emptyset \rangle}{\langle \alpha_x@a^\bullet, i, \_, \_, l, \Lambda^1, \ldots, \Lambda^d \rangle}$$

$\mathsf{Subst}(\alpha_x@a, \beta_y)$
$\mathsf{Filter}(\{\beta_y\}, \Lambda_1^1, \ldots, \Lambda_1^{d-1}) = \Lambda^1, \ldots, \Lambda^d$

No Adjoin:

$$\frac{\langle \alpha_x@a^\circ, i, j, k, l, \Lambda^1, \ldots, \Lambda^d \rangle}{\langle \alpha_x@a^\bullet, i, j, k, l, \Lambda^1, \ldots, \Lambda^d \rangle}$$

Figure 4: Axioms and inference rules for the CKY algorithm for restricted NS-MCTAG with a restriction of d.

history at the limit of the permissible delay must be empty for the operation to succeed.

6 Restricted V-TAG

A Vector-TAG (V-TAG) (Rambow, 1994) is similar to an MCTAG in that the elementary structures are sets (or vectors) of TAG trees. A derivation in a V-TAG is defined as in TAG. There is no locality requirement or other restriction on adjunction except that if one tree from a vector is used in a derivation, all trees from that vector must be used in the derivation. The trees in a vector may be connected by dominance links between the foot nodes of auxiliary trees and any node in other trees in the vector. All adjunctions must respect the dominance relations in that a node η_1 that dominates a node η_2 must appear on the path from η_2 to the root of the derived tree. The definition of V-TAG is very similar to that of non-local MCTAG as defined by Weir (1988) except that in non-local MCTAG all trees from a tree set are required to adjoin simultaneously.

Restricted V-TAG constrains V-TAG in several ways. First, the dominance chain in each elementary tree vector is required to define a total order over the trees in the vector. This means there is a single **base tree** in each vector. Note also that all trees other than the base tree must be auxiliary trees in order to dominate other trees in the vector. The base tree may be either an initial tree or an auxiliary tree. Second, a restricted V-TAG has a restriction level, d, that determines the largest derivational distance that may exists between the base tree and the highest tree in a tree vector in a derivation. Restricted V-TAG differs from restricted NS-MCTAG in one important respect: the dominance requirements of restricted V-TAG require that trees from the same

set must appear along a single path in the derived tree, whereas in restricted NS-MCTAG trees from the same set need not adhere to any dominance relationship in the derived tree.

A CKY-style parser for restricted V-TAG with restriction level d is given in Figure 5. Parsing is similar to delayed TL-MCTAG in that we have a set of histories for each restriction level. However, because of the total order over trees in a vector, the parser only needs to maintain the identity of the highest tree from a vector that has been used in the derivation along with its distance from the base tree from that vector. The Filter side condition accordingly expunges trees that are the top tree in the dominance chain of their tree vector. The side conditions for the Adjoin non-base rule enforce that the dominance constraints are satisfied and that the derivational distance from the base of a tree vector to its currently highest adjoined tree is maintained accurately. We note that in order to allow a non-total ordering of the trees in a vector we would simply have to record all trees in a tree vector in the histories as is done in the delayed TL-MCTAG parser.

7 Delayed TL-MCTAG

Chiang and Scheffler (2008) introduce the delayed TL-MCTAG formalism which makes use of a derivational distance restriction in a somewhat different way. Rather than restricting the absolute distance between the trees of a set and their nearest common ancestor, given a node α in a derivation tree, delayed TL-MCTAG restricts the number of tree sets that are not fully dominated by α. Borrowing directly from Chiang and Scheffler (2008), Figure 7 gives two examples.

Parsing for delayed TL-MCTAG is not discussed by Chiang and Scheffler (2008) but can be accomplished using a similar CKY-style strategy as in the two parsers above. We present a parser in Figure 6. Rather than keeping histories that record derivational distance, we keep an active delay list for each item that records the delays that are active (by recording the identities of the trees that have adjoined) for the tree of which the current node is a part. At the root of each tree the active delay list is filtered using the Filter side condition to remove all tree sets that are fully dominated and the resulting

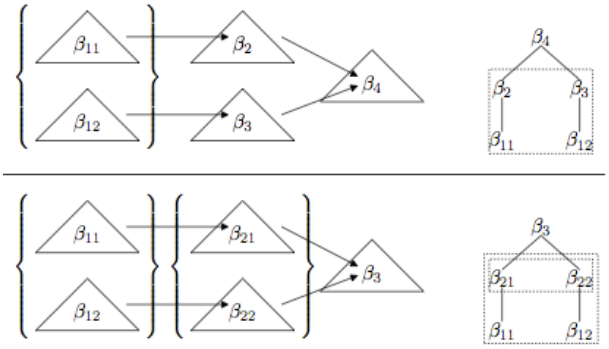


Figure 7: Examples of 1-delay (top) and 2-delay (bottom) taken from Chiang and Scheffler (2008). The delays are marked with dashed boxes on the derivation trees.

list is checked using the Size to ensure that it contains no more than d distinct tree sets where d is the specified delay for the grammar. The active delays for a given tree are passed to its derivational parent when it adjoins or substitutes.

Delayed TL-MCTAG differs from both of the previous formalisms in that it places no constraint on the length of a delay. On the other hand while the previous formalisms allow unlimited short delays to be pending at the same time, in delayed TL-MCTAG, only a restricted number of delays may be active at once. Similar to restricted V-TAG, there is no simultaneity requirement, so a tree may have another tree from the same set as an ancestor.

8 Complexity

The complexity of the restricted NS-MCTAG and restricted V-TAG parsers presented above depends on the number of possible histories that may appear in an item. For each step of derivational distance permitted between trees of the same set, the corresponding history permits many more entries. History Λ^1 may contain trees that have adjoined into the same tree as the node of the current item. The number of entries is therefore limited by the number of adjunction sites in that tree, which is in turn limited by the number of nodes in that tree. We will call the maximum number of nodes in a tree in the grammar t. Theoretically, any tree in the grammar could adjoin at any of these adjunction sites, meaning that the number of possible values for each entry in the history is bounded by the size of the grammar $|G|$. Thus the size of Λ^1 is $O(|G|^t)$. For Λ^2 the en-

Unary Complete

$$\frac{\langle \alpha_x @ (a \cdot 1)^\bullet, i, j, k, l, \Lambda^1, \ldots, \Lambda^d \rangle}{\langle \alpha_x @ a^\circ, i, j, k, l, \Lambda^1, \ldots, \Lambda^d \rangle} \qquad \alpha_x @ (a \cdot 2) \text{ undefined}$$

Binary Complete

$$\frac{\langle \alpha_x @ (a \cdot 1)^\bullet, i, j, k, l, \Lambda_1^1, \ldots, \Lambda_1^d \rangle \langle \alpha_x @ (a \cdot 2)^\bullet, l, j', k', m, \Lambda_2^1, \ldots, \Lambda_2^d \rangle}{\langle \alpha_x @ a^\circ, i, j \cup j', k \cup k', m, \Lambda_1^1 \cup \Lambda_2^1, \ldots, \Lambda_1^d \cup \Lambda_2^d \rangle}$$

Adjoin base:

$$\frac{\langle \beta_1 @ \varepsilon^\bullet, i, p, q, l, \Lambda_1^1, \ldots, \Lambda_1^{d-1}, \emptyset \rangle \langle \alpha_x @ a^\circ, p, j, k, q, \Lambda_2^1, \ldots, \Lambda_2^d \rangle}{\langle \alpha_x @ a^\bullet, i, j, k, l, \Lambda^1, \ldots, \Lambda^d \rangle}$$

$\mathsf{Adj}(\alpha_x @ a, \beta_1)$
$\mathsf{Filter}(\Lambda_2^1 \cup \{\beta_1\}, \Lambda_2^2 \cup \Lambda_1^1, \ldots, \Lambda_2^d \cup \Lambda_1^{d-1}) = \Lambda^1, \ldots, \Lambda^d$

Adjoin non-base:

$$\frac{\langle \beta_y @ \varepsilon^\bullet, i, p, q, l, \Lambda_1^1, \ldots, \Lambda_1^{d-1}, \emptyset \rangle \langle \alpha_x @ a^\circ, p, j, k, q, \Lambda_2^1, \ldots, \Lambda_2^d \rangle}{\langle \alpha_x @ a^\bullet, i, j, k, l, \Lambda^1, \ldots, \Lambda^d \rangle}$$

for unique Λ_2^i s.t. $\beta_{y-1} \in \Lambda_2^i, \Lambda_{2'}^i = (\Lambda_2^i \cup \Lambda_1^{i-1} \cup \{\beta_y\}) - \{\beta_{y-1}\}$
for Λ_2^i s.t. $\beta_{y-1} \notin \Lambda_2^i, \Lambda_{2'}^i = \Lambda_2^i \cup \Lambda_1^{i-1}$

$\mathsf{Adj}(\alpha_x @ a, \beta_y)$
$\mathsf{Filter}(\Lambda_{2'}^1, \Lambda_{2'}^2 \cup \Lambda_1^1, \ldots, \Lambda_{2'}^d \cup \Lambda_1^{d-1}) = \Lambda^1, \ldots, \Lambda^d$

Substitute:

$$\frac{\langle \beta_1 @ \varepsilon^\bullet, i, \_, \_, l, \Lambda_1^1, \ldots, \Lambda_1^{d-1}, \emptyset \rangle}{\langle \alpha_x @ a^\bullet, i, \_, \_, l, \Lambda^1, \ldots, \Lambda^d \rangle}$$

$\mathsf{Subst}(\alpha_x @ a, \beta_1)$
$\mathsf{Filter}(\{\beta_1\}, \Lambda_1^1, \ldots, \Lambda_1^{d-1}) = \Lambda^1, \ldots, \Lambda^d$

No Adjoin:

$$\frac{\langle \alpha_x @ a^\circ, i, j, k, l, \Lambda^1, \ldots, \Lambda^d \rangle}{\langle \alpha_x @ a^\bullet, i, j, k, l, \Lambda^1, \ldots, \Lambda^d \rangle}$$

Figure 5: Inference rules for the CKY algorithm for restricted V-TAG with a restriction of d. Item form, goal item and axioms are omitted because they are identical to those in restricted NS-MCTAG parser.

tries correspond to tree that have adjoined into a tree that has adjoined into the tree of the current item. Thus, for each of the t trees that may have adjoined at a derivational distance of one, there are t more trees that may have adjoined at a derivational distance of two. The size of Λ^2 is therefore $|G|^{t^2}$. The combined size of the histories for a grammar with a delay or restriction of d is therefore $O(|G|^{\sum_{i=1}^{d} t^d})$. Replacing the sum with its closed form solution, we have $O(|G|^{\frac{t^{d+1}-1}{t-1}-1})$ histories.

Using the reasoning about the size of the histories given above, the restricted NS-MCTAG parser presented here has a complexity of $O(n^6 |G|^{1+\frac{t^{d+1}-1}{t-1}})$, where t is as defined above and d is the limit on delay of adjunction. For a tree-local MCTAG, the complexity reduces to $O(n^6 |G|^{2+t})$. For the linguistic applications that motivate this chapter no delay greater than two is needed, resulting in a complexity of $O(n^6 |G|^{2+t+t^2})$.

The same complexity analysis applies for restricted V-TAG. However, we can provide a somewhat tighter bound by noting that the **rank**, r, of the grammar—how many tree sets adjoin in a single tree—and the **fan out**, f of the grammar—how many trees may be in a single tree set—are limited by t. That is, a complete derivation containing $|\mathcal{D}|$ tree sets can contain no more than $t\,|\mathcal{D}|$ individual trees and also no more than $rf\,|\mathcal{D}|$ individual trees. In the restricted V-TAG algorithm we maintain only one tree from a tree set in the history at a time, so rather than maintaining $O(t)$ entries in each history, we only need to maintain the smaller $O(r)$ entries.

The complexity of the delayed TL-MCTAG parser depends on the number of possible active delay lists. As above, each delay list may have a maximum of t entries for trees that adjoin directly into it. The restriction on the number of active delays means that the active delay lists passed up from these child nodes at the point of adjunction or substitution can have size no more than d. This results in an additional $td(f-1)$ possible entries in the active de-

Goal Item:

$$\langle \alpha_0 @ \varepsilon^\bullet, 0, \_, \_, n, \emptyset, \ldots, \emptyset \rangle \qquad \begin{array}{l} \mathsf{Init}(\alpha_1) \\ \mathsf{Label}(\alpha_0 @ \varepsilon) = S \\ |\alpha| = 1 \end{array}$$

Terminal Axiom

$$\langle \alpha_x @ a^\bullet, i-1, \_, \_, i, \emptyset, \ldots, \{\alpha_x\} \rangle \qquad \mathsf{Label}(\alpha_x @ a) = w_i$$

Empty Axiom

$$\langle \alpha_x @ a^\bullet, i, \_, \_, i, \emptyset, \ldots, \{\alpha_x\} \rangle \qquad \mathsf{Label}(\alpha_x @ a) = \varepsilon$$

Foot Axiom

$$\langle \alpha_x @ \mathsf{Ft}(\alpha_x)^\circ, p, p, q, q, \emptyset, \ldots, \{\alpha_x\} \rangle \qquad \mathsf{Aux}(\alpha_x)$$

Unary Complete

$$\frac{\langle \alpha_x @ (a \cdot 1)^\bullet, i, j, k, l, \Lambda \rangle}{\langle \alpha_x @ a^\circ, i, j, k, l, \Lambda \rangle} \qquad \alpha_x @ (a \cdot 2) \text{ undefined}$$

Binary Complete

$$\frac{\langle \alpha_x @ (a \cdot 1)^\bullet, i, j, k, l, \Lambda_1 \rangle \langle \alpha_x @ (a \cdot 2)^\bullet, l, j', k', m, \Lambda_2 \rangle}{\langle \alpha_x @ a^\circ, i, j \cup j', k \cup k', m, \Lambda_1 \cup \Lambda_2 \rangle}$$

Adjoin:

$$\frac{\langle \beta_y @ \varepsilon^\bullet, i, p, q, l, \Lambda_\beta \rangle \langle \alpha_x @ a^\circ, p, j, k, q, \Lambda_\alpha \rangle}{\langle \alpha_x @ a^\bullet, i, j, k, l, \Lambda'_\beta \cup \Lambda_\alpha \rangle} \qquad \begin{array}{l} \mathsf{Adj}(\alpha_x @ a, \beta_y) \\ \mathsf{Filter}(\Lambda_\beta, \Lambda'_\beta) \\ \mathsf{Size}(\Lambda'_\beta) \leq d \end{array}$$

Substitute:

$$\frac{\langle \beta_y @ \varepsilon^\bullet, i, \_, \_, l, \Lambda_\beta \rangle}{\langle \alpha_x @ a^\bullet, i, \_, \_, l, \Lambda'_\beta \cup \{\alpha_x\} \rangle} \qquad \begin{array}{l} \mathsf{Subst}(\alpha_x @ a, \beta_y) \\ \mathsf{Filter}(\Lambda_\beta, \Lambda'_\beta) \\ \mathsf{Size}(\Lambda'_\beta) \leq d \end{array}$$

No Adjoin:

$$\frac{\langle \alpha_x @ a^\circ, i, j, k, l, \Lambda \rangle}{\langle \alpha_x @ a^\bullet, i, j, k, l, \Lambda \rangle}$$

Figure 6: Axioms and inference rules for the CKY algorithm for delayed TL-MCTAG with a delay of d.

lay list, giving a total number of active delay lists of $O(|G|^{t(1+d(f-1))})$. Thus the complexity of the parser is $O(n^6 \, |G|^{2+t(1+d(f-1))})$.

9 Conclusion

Each of the formalisms presented above extends the flexibility of MCTAG beyond that of TL-MCTAG while maintaining, as we have shown herein, complexity much less than that of SL-MCTAG. All three formalisms permit modeling of flexible composition (because they permit one member of a tree set to be a derivational ancestor of another tree in the same set), at least restricted NS-MCTAG and restricted V-TAG permit analyses of scrambling, and all three permit analyses of the various challenging semantic constructions mentioned in the introduction. We conclude that extending locality by constraining derivational distance may be an effective way to add flexibility to MCTAG without losing computational tractability.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. BCS-0827979.

References

David Chiang and Tatjana Scheffler. 2008. Flexible composition and delayed tree-locality. In *The Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+9)*.

Aravind K. Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, pages 69–124. Springer.

Laura Kallmeyer and Maribel Romero. 2004. LTAG semantics with semantic unification. In *Proceedings of the 7th International Workshop on Tree-Adjoining Grammars and Related Formalisms (TAG+7)*, pages 155–162, Vancouver, May.

Laura Kallmeyer. 2007. A declarative characterization of different types of multicomponent tree adjoining grammars. In Andreas Witt Georg Rehm and Lothar Lemnitzer, editors, *Datenstrukturen für linguistische Ressourcen und ihre Anwendungen*, pages 111–120.

T. Kasami. 1965. An efficient recognition and syntax algorithm for context-free languages. Technical Report AF-CRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

Rebecca Nesson and Stuart M. Shieber. 2006. Simpler TAG semantics through synchronization. In *Proceedings of the 11th Conference on Formal Grammar*, Malaga, Spain, 29–30 July.

Rebecca Nesson, Giorgio Satta, and Stuart M. Shieber. 2008. Complexity, parsing, and factorization of tree-local multi-component tree-adjoining grammar. Technical report, Harvard University.

Owen Rambow. 1994. *Formal and computational aspects of natural language syntax*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Maribel Romero, Laura Kallmeyer, and Olga Babko-Malaya. 2004. LTAG semantics for questions. In *Proceedings of the 7th International Workshop on Tree-Adjoining Grammars and Related Formalisms (TAG+7)*, pages 186–193, Vancouver, May.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36, July–August. Also available as cmp-lg/9404008.

Anders Søgaard, Timm Lichte, and Wolfgang Maier. 2007. On the complexity of linguistically motivated extensions of tree-adjoining grammar. In *Recent Advances in Natural Language Processing 2007*.

K. Vijay-Shanker. 1987. A study of tree-adjoining grammars. PhD Thesis, Department of Computer and Information Science, University of Pennsylvania.

David Weir. 1988. Characterizing mildly context-sensitive grammar formalisms. PhD Thesis, Department of Computer and Information Science, University of Pennsylvania.

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

Improving Unsupervised Dependency Parsing with Richer Contexts and Smoothing

William P. Headden III, Mark Johnson, David McClosky

Brown Laboratory for Linguistic Information Processing (BLLIP)

Brown University

Providence, RI 02912

{headdenw,mj,dmcc}@cs.brown.edu

Abstract

Unsupervised grammar induction models tend to employ relatively simple models of syntax when compared to their supervised counterparts. Traditionally, the unsupervised models have been kept simple due to tractability and data sparsity concerns. In this paper, we introduce basic valence frames and lexical information into an unsupervised dependency grammar inducer and show how this additional information can be leveraged via smoothing. Our model produces state-of-the-art results on the task of unsupervised grammar induction, improving over the best previous work by almost 10 percentage points.

1 Introduction

The last decade has seen great strides in statistical natural language parsing. Supervised and semi-supervised methods now provide highly accurate parsers for a number of languages, but require training from corpora hand-annotated with parse trees. Unfortunately, manually annotating corpora with parse trees is expensive and time consuming so for languages and domains with minimal resources it is valuable to study methods for parsing without requiring annotated sentences.

In this work, we focus on unsupervised dependency parsing. Our goal is to produce a directed graph of dependency relations (e.g. Figure 1) where each edge indicates a head-argument relation. Since the task is unsupervised, we are not given any examples of correct dependency graphs and only take words and their parts of speech as input. Most of the recent work in this area (Smith, 2006; Cohen et al., 2008) has focused on variants of the

Figure 1: Example dependency parse.

Dependency Model with Valence (DMV) by Klein and Manning (2004). DMV was the first unsupervised dependency grammar induction system to achieve accuracy above a right-branching baseline. However, DMV is not able to capture some of the more complex aspects of language. Borrowing some ideas from the supervised parsing literature, we present two new models: Extended Valence Grammar (EVG) and its lexicalized extension (L-EVG). The primary difference between EVG and DMV is that DMV uses valence information to determine the number of arguments a head takes but not their categories. In contrast, EVG allows different distributions over arguments for different valence slots. L-EVG extends EVG by conditioning on lexical information as well. This allows L-EVG to potentially capture subcategorizations. The downside of adding additional conditioning events is that we introduce data sparsity problems. Incorporating more valence and lexical information increases the number of parameters to estimate. A common solution to data sparsity in supervised parsing is to add smoothing. We show that smoothing can be employed in an unsupervised fashion as well, and show that mixing DMV, EVG, and L-EVG together produces state-of-the-art results on this task. To our knowledge, this is the first time that grammars with differing levels of detail have been successfully combined for unsupervised dependency parsing.

A brief overview of the paper follows. In Section 2, we discuss the relevant background. Section 3 presents how we will extend DMV with additional

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 101–109,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

features. We describe smoothing in an unsupervised context in Section 4. In Section 5, we discuss search issues. We present our experiments in Section 6 and conclude in Section 7.

2 Background

In this paper, the observed variables will be a corpus of n sentences of text $\mathbf{s} = s_1 \dots s_n$, and for each word s_{ij} an associated part-of-speech τ_{ij}. We denote the set of all words as V_w and the set of all parts-of-speech as V_τ. The hidden variables are parse trees $\mathbf{t} = t_1 \dots t_n$ and parameters $\bar{\theta}$ which specify a distribution over $\mathbf{t}$. A dependency tree t_i is a directed acyclic graph whose nodes are the words in s_i. The graph has a single incoming edge for each word in each sentence, except one called the *root* of t_i. An edge from word i to word j means that word j is an *argument* of word i or alternatively, word i is the *head* of word j. Note that each word token may be the argument of at most one head, but a head may have several arguments.

If parse tree t_i can be drawn on a plane above the sentence with no crossing edges, it is called *projective*. Otherwise it is *nonprojective*. As in previous work, we restrict ourselves to projective dependency trees. The dependency models in this paper will be formulated as a particular kind of Probabilistic Context Free Grammar (PCFG), described below.

2.1 Tied Probabilistic Context Free Grammars

In order to perform smoothing, we will find useful a class of PCFGs in which the probabilities of certain rules are required to be the same. This will allow us to make independence assumptions for smoothing purposes without losing information, by giving analogous rules the same probability.

Let $G = (\mathcal{N}, \mathcal{T}, S, \mathcal{R}, \theta)$ be a Probabilistic Context Free Grammar with nonterminal symbols $\mathcal{N}$, terminal symbols $\mathcal{T}$, start symbol $S \in \mathcal{N}$, set of productions $\mathcal{R}$ of the form $N \to \beta$, $N \in \mathcal{N}, \beta \in (\mathcal{N} \cup \mathcal{T})^*$. Let $\mathcal{R}_N$ indicate the subset of $\mathcal{R}$ whose left-hand sides are N. θ is a vector of length $|\mathcal{R}|$, indexed by productions $N \to \beta \in \mathcal{R}$. $\theta_{N \to \beta}$ specifies the probability that N rewrites to β. We will let θ_N indicate the subvector of θ corresponding to $\mathcal{R}_N$.

A tied PCFG constrains a PCFG G with a tying relation, which is an equivalence relation over rules that satisfies the following properties:

1. Tied rules have the same probability.
2. Rules expanding the same nonterminal are never tied.
3. If $N_1 \to \beta_1$ and $N_2 \to \beta_2$ are tied then the tying relation defines a one-to-one mapping between rules in $\mathcal{R}_{N_1}$ and $\mathcal{R}_{N_2}$, and we say that N_1 and N_2 are tied nonterminals.

As we see below, we can estimate tied PCFGs using standard techniques. Clearly, the tying relation also defines an equivalence class over nonterminals. The tying relation allows us to formulate the distributions over trees in terms of rule equivalence classes and nonterminal equivalence classes. Suppose $\bar{\mathcal{R}}$ is the set of rule equivalence classes and $\bar{\mathcal{N}}$ is the set of nonterminal equivalence classes. Since all rules in an equivalence class $\bar{r}$ have the same probability (condition 1), and since all the nonterminals in an equivalence class $\bar{N} \in \bar{\mathcal{N}}$ have the same distribution over rule equivalence classes (condition 1 and 3), we can define the set of rule equivalence classes $\bar{\mathcal{R}}_{\bar{N}}$ associated with a nonterminal equivalence class $\bar{N}$, and a vector $\bar{\theta}$ of probabilities, indexed by rule equivalence classes $\bar{r} \in \bar{\mathcal{R}}$. $\bar{\theta}_{\bar{N}}$ refers to the subvector of $\bar{\theta}$ associated with nonterminal equivalence class $\bar{N}$, indexed by $\bar{r} \in \bar{\mathcal{R}}_{\bar{N}}$. Since rules in the same equivalence class have the same probability, we have that for each $r \in \bar{r}$, $\theta_r = \bar{\theta}_{\bar{r}}$.

Let $f(\mathbf{t}, r)$ denote the number of times rule r appears in tree $\mathbf{t}$, and let $f(\mathbf{t}, \bar{r}) = \sum_{r \in \bar{r}} f(\mathbf{t}, r)$. We see that the complete data likelihood is

$$P(\mathbf{s}, \mathbf{t}|\theta) = \prod_{\bar{r} \in \bar{\mathcal{R}}} \prod_{r \in \bar{r}} \theta_r^{f(\mathbf{t},r)} = \prod_{\bar{r} \in \bar{\mathcal{R}}} \bar{\theta}_{\bar{r}}^{f(\mathbf{t},\bar{r})}$$

That is, the likelihood is a product of multinomials, one for each nonterminal equivalence class, and there are no constraints placed on the parameters of these multinomials besides being positive and summing to one. This means that all the standard estimation methods (e.g. Expectation Maximization, Variational Bayes) extend directly to tied PCFGs.

Maximum likelihood estimation provides a point estimate of $\bar{\theta}$. However, often we want to incorporate information about $\bar{\theta}$ by modeling its *prior* distribution. As a prior, for each $\bar{N} \in \bar{\mathcal{N}}$ we will specify a

Dirichlet distribution over $\bar{\theta}_{\bar{N}}$ with hyperparameters $\alpha_{\bar{N}}$. The Dirichlet has the density function:

$$P(\bar{\theta}_{\bar{N}}|\alpha_{\bar{N}}) = \frac{\Gamma(\sum_{\bar{r}\in\bar{\mathcal{R}}_{\bar{N}}} \alpha_{\bar{r}})}{\prod_{\bar{r}\in\bar{\mathcal{R}}_{\bar{N}}} \Gamma(\alpha_{\bar{r}})} \prod_{\bar{r}\in\bar{\mathcal{R}}_{\bar{N}}} \bar{\theta}_{\bar{r}}^{\alpha_{\bar{r}}-1},$$

Thus the prior over $\bar{\theta}$ is a product of Dirichlets,which is *conjugate* to the PCFG likelihood function (Johnson et al., 2007). That is, the posterior $P(\bar{\theta}|\mathbf{s},\mathbf{t},\alpha)$ is also a product of Dirichlets, also factoring into a Dirichlet for each nonterminal $\bar{N}$, where the parameters $\alpha_{\bar{r}}$ are augmented by the number of times rule $\bar{r}$ is observed in tree $\mathbf{t}$:

$$\begin{aligned} P(\bar{\theta}|\mathbf{s},\mathbf{t},\alpha) &\propto P(\mathbf{s},\mathbf{t}|\bar{\theta})P(\bar{\theta}|\alpha) \\ &\propto \prod_{\bar{r}\in\bar{\mathcal{R}}} \bar{\theta}_{\bar{r}}^{f(\mathbf{t},\bar{r})+\alpha_{\bar{r}}-1} \end{aligned}$$

We can see that $\alpha_{\bar{r}}$ acts as a pseudocount of the number of times $\bar{r}$ is observed prior to $\mathbf{t}$.

To make use of this prior, we use the Variational Bayes (VB) technique for PCFGs with Dirichlet Priors presented by Kurihara and Sato (2004). VB estimates a distribution over $\bar{\theta}$. In contrast, Expectation Maximization estimates merely a point estimate of $\bar{\theta}$. In VB, one estimates $Q(\mathbf{t},\bar{\theta})$, called the variational distribution, which approximates the posterior distribution $P(\mathbf{t},\bar{\theta}|\mathbf{s},\alpha)$ by minimizing the KL divergence of P from Q. Minimizing the KL divergence, it turns out, is equivalent to maximizing a lower bound $\mathcal{F}$ of the log marginal likelihood $\log P(\mathbf{s}|\alpha)$.

$$\log P(\mathbf{s}|\alpha) \geq \sum_{\mathbf{t}} \int_{\bar{\theta}} Q(\mathbf{t},\bar{\theta}) \log \frac{P(\mathbf{s},\mathbf{t},\bar{\theta}|\alpha)}{Q(\mathbf{t},\bar{\theta})} = \mathcal{F}$$

The negative of the lower bound, $-\mathcal{F}$, is sometimes called the *free energy*.

As is typical in variational approaches, Kurihara and Sato (2004) make certain independence assumptions about the hidden variables in the variational posterior, which will make estimating it simpler. It factors $Q(\mathbf{t},\bar{\theta}) = Q(\mathbf{t})Q(\bar{\theta}) = \prod_{i=1}^{n} Q_i(t_i) \prod_{\bar{N}\in\bar{\mathcal{N}}} Q(\bar{\theta}_{\bar{N}})$. The goal is to recover $Q(\bar{\theta})$, the estimate of the posterior distribution over parameters and $Q(t)$, the estimate of the posterior distribution over trees. Finding a local maximum of $\mathcal{F}$ is done via an alternating maximization of $Q(\bar{\theta})$ and $Q(\mathbf{t})$. Kurihara and Sato (2004) show that each $Q(\bar{\theta}_{\bar{N}})$ is a Dirichlet distribution with parameters $\hat{\alpha}_r = \alpha_r + E_{Q(\mathbf{t})} f(\mathbf{t},r)$.

2.2 Split-head Bilexical CFGs

In the sections that follow, we frame various dependency models as a particular variety of CFGs known as split-head bilexical CFGs (Eisner and Satta, 1999). These allow us to use the fast Eisner and Satta (1999) parsing algorithm to compute the expectations required by VB in $O(m^3)$ time (Eisner and Blatz, 2007; Johnson, 2007) where m is the length of the sentence.[1]

In the split-head bilexical CFG framework, each nonterminal in the grammar is annotated with a terminal symbol. For dependency grammars, these annotations correspond to words and/or parts-of-speech. Additionally, split-head bilexical CFGs require that each word s_{ij} in sentence s_i is represented in a split form by two terminals called its left part s_{ijL} and right part s_{ijR}. The set of these parts constitutes the terminal symbols of the grammar. This split-head property relates to a particular type of dependency grammar in which the left and right dependents of a head are generated independently. Note that like CFGs, split-head bilexical CFGs can be made probabilistic.

2.3 Dependency Model with Valence

The most successful recent work on dependency induction has focused on the Dependency Model with Valence (DMV) by Klein and Manning (2004). DMV is a generative model in which the head of the sentence is generated and then each head recursively generates its left and right dependents. The arguments of head H in direction d are generated by repeatedly deciding whether to generate another new argument or to stop and then generating the argument if required. The probability of deciding whether to generate another argument is conditioned on H, d and whether this would be the first argument (this is the sense in which it models valence). When DMV generates an argument, the part-of-speech of that argument A is generated given H and d.

[1] Efficiently parsable versions of split-head bilexical CFGs for the models described in this paper can be derived using the fold-unfold grammar transform (Eisner and Blatz, 2007; Johnson, 2007).

Rule	Description
$S \rightarrow Y_H$	Select H as root
$Y_H \rightarrow L_H\ R_H$	Move to split-head representation
$L_H \rightarrow H_L$	STOP $\mid dir = L, head = H, val = 0$
$L_H \rightarrow L^1_H$	CONT $\mid dir = L, head = H, val = 0$
$L'_H \rightarrow H_L$	STOP $\mid dir = L, head = H, val = 1$
$L'_H \rightarrow L^1_H$	CONT $\mid dir = L, head = H, val = 1$
$L^1_H \rightarrow Y_A\ L'_H$	Arg $A \mid dir = L, head = H$

Figure 2: Rule schema for DMV. For brevity, we omit the portion of the grammar that handles the right arguments since they are symmetric to the left (all rules are the same except for the attachment rule where the RHS is reversed). $val \in \{0, 1\}$ indicates whether we have made any attachments.

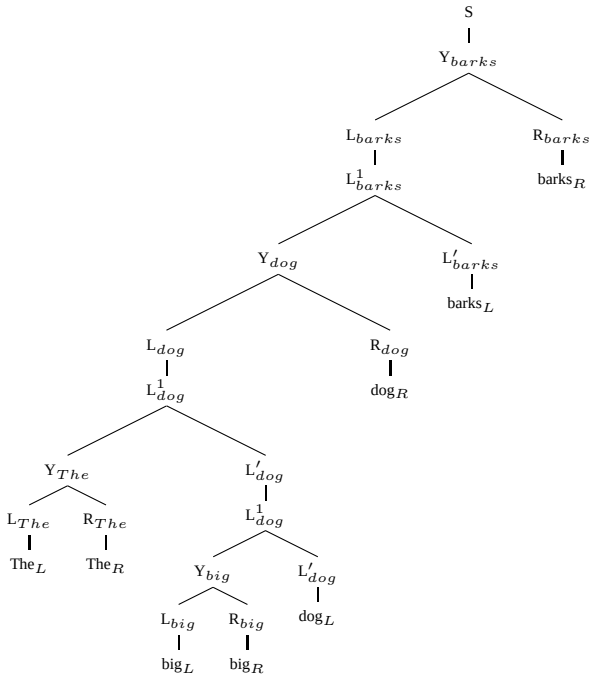


Figure 3: DMV split-head bilexical CFG parse of "The big dog barks."

The grammar schema for this model is shown in Figure 2. The first rule generates the root of the sentence. Note that these rules are for $\forall H, A \in V_\tau$ so there is an instance of the first schema rule for each part-of-speech. Y_H splits words into their left and right components. L_H encodes the stopping decision given that we have not generated any arguments so far. L'_H encodes the same decision after generating one or more arguments. L^1_H represents the distribution over left attachments. To extract dependency relations from these parse trees, we scan for attachment rules (e.g., $L^1_H \rightarrow Y_A\ L'_H$) and record that A depends on H. The schema omits the rules for right arguments since they are symmetric. We show a parse of "The big dog barks" in Figure 3.[2]

Much of the extensions to this work have focused on estimation procedures. Klein and Manning (2004) use Expectation Maximization to estimate the model parameters. Smith and Eisner (2005) and Smith (2006) investigate using Contrastive Estimation to estimate DMV. Contrastive Estimation maximizes the conditional probability of the observed sentences given a neighborhood of similar unseen sequences. The results of this approach vary widely based on regularization and neighborhood, but often outperforms EM.

[2]Note that our examples use words as leaf nodes but in our unlexicalized models, the leaf nodes are in fact parts-of-speech.

Smith (2006) also investigates two techniques for maximizing likelihood while incorporating the locality bias encoded in the harmonic initializer for DMV. One technique, skewed deterministic annealing, ameliorates the local maximum problem by flattening the likelihood and adding a bias towards the Klein and Manning initializer, which is decreased during learning. The second technique is structural annealing (Smith and Eisner, 2006; Smith, 2006) which penalizes long dependencies initially, gradually weakening the penalty during estimation. If hand-annotated dependencies on a held-out set are available for parameter selection, this performs far better than EM; however, performing parameter selection on a held-out set without the use of gold dependencies does not perform as well.

Cohen et al. (2008) investigate using Bayesian Priors with DMV. The two priors they use are the Dirichlet (which we use here) and the Logistic Normal prior, which allows the model to capture correlations between different distributions. They initialize using the harmonic initializer of Klein and Manning (2004). They find that the Logistic Normal distribution performs much better than the Dirichlet with this initialization scheme.

Cohen and Smith (2009), investigate (concur-

Rule	Description
$S \rightarrow Y_H$	Select H as root
$Y_H \rightarrow L_H\ R_H$	Move to split-head representation
$L_H \rightarrow H_L$	STOP $\mid dir = L, head = H, val = 0$
$L_H \rightarrow L'_H$	CONT $\mid dir = L, head = H, val = 0$
$L'_H \rightarrow L^1_H$	STOP $\mid dir = L, head = H, val = 1$
$L'_H \rightarrow L^2_H$	CONT $\mid dir = L, head = H, val = 1$
$L^2_H \rightarrow Y_A\ L'_H$	Arg $A \mid dir = L, head = H, val = 1$
$L^1_H \rightarrow Y_A\ H_L$	Arg $A \mid dir = L, head = H, val = 0$

Figure 4: Extended Valence Grammar schema. As before, we omit rules involving the right parts of words. In this case, $val \in \{0, 1\}$ indicates whether we are generating the nearest argument (0) or not (1).

rently with our work) an extension of this, the Shared Logistic Normal prior, which allows different PCFG rule distributions to share components. They use this machinery to investigate smoothing the attachment distributions for (nouns/verbs), and for learning using multiple languages.

3 Enriched Contexts

DMV models the distribution over arguments identically without regard to their order. Instead, we propose to distinguish the distribution over the argument nearest the head from the distribution of subsequent arguments. [3]

Consider the following changes to the DMV grammar (results shown in Figure 4). First, we will introduce the rule $L^2_H \rightarrow Y_A\ L'_H$ to denote the decision of what argument to generate for positions not nearest to the head. Next, instead of having L'_H expand to H_L or L^1_H, we will expand it to L^1_H (attach to nearest argument and stop) or L^2_H (attach to non-nearest argument and continue). We call this the *Extended Valence Grammar* (EVG).

As a concrete example, consider the phrase "the big hungry dog" (Figure 5). We would expect that distribution over the nearest left argument for "dog" to be different than farther left arguments. The figure shows that EVG allows these two distributions to be different (nonterminals L^2_{dog} and L^1_{dog}) whereas DMV forces them to be equivalent (both use L^1_{dog} as the nonterminal).

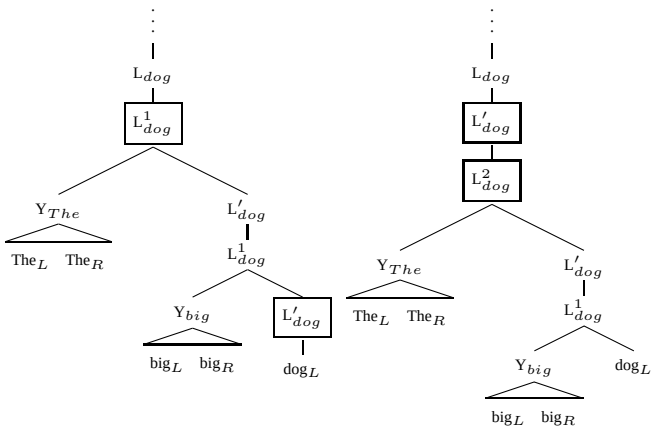


Figure 5: An example of moving from DMV to EVG for a fragment of "The big dog." Boxed nodes indicate changes. The key difference is that EVG distinguishes between the distributions over the argument nearest the head (*big*) from arguments farther away (*The*).

3.1 Lexicalization

All of the probabilistic models discussed thus far have incorporated only part-of-speech information (see Footnote 2). In supervised parsing of both dependencies and constituency, lexical information is critical (Collins, 1999). We incorporate lexical information into EVG (henceforth L-EVG) by extending the distributions over argument parts-of-speech A to condition on the head word h in addition to the head part-of-speech H, direction d and argument position v. The argument word a distribution is merely conditioned on part-of-speech A; we leave refining this model to future work.

In order to incorporate lexicalization, we extend the EVG CFG to allow the nonterminals to be annotated with both the word and part-of-speech of the head. We first remove the old rules $Y_H \rightarrow L_H\ R_H$ for each $H \in V_\tau$. Then we mark each nonterminal which is annotated with a part-of-speech as also annotated with its head, with a single exception: Y_H. We add a new nonterminal $Y_{H,h}$ for each $H \in V_\tau, h \in V_w$, and the rules $Y_H \rightarrow Y_{H,h}$ and $Y_{H,h} \rightarrow L_{H,h}\ R_{H,h}$. The rule $Y_H \rightarrow Y_{H,h}$ corresponds to selecting the word, given its part-of-speech.

[3] McClosky (2008) explores this idea further in an unsmoothed grammar.

4 Smoothing

In supervised estimation one common smoothing technique is *linear interpolation*, (Jelinek, 1997). This section explains how linear interpolation can be represented using a PCFG with tied rule probabilities, and how one might estimate smoothing parameters in an unsupervised framework.

In many probabilistic models it is common to estimate the distribution of some event x conditioned on some set of context information $P(x|N_{(1)} \dots N_{(k)})$ by smoothing it with less complicated conditional distributions. Using linear interpolation we model $P(x|N_{(1)} \dots N_{(k)})$ as a weighted average of two distributions $\lambda_1 P_1(x|N_{(1)}, \dots, N_{(k)}) + \lambda_2 P_2(x|N_{(1)}, \dots, N_{(k-1)})$, where the distribution P_2 makes an independence assumption by dropping the conditioning event $N_{(k)}$.

In a PCFG a nonterminal N can encode a collection of conditioning events $N_{(1)} \dots N_{(k)}$, and θ_N determines a distribution conditioned on $N_{(1)} \dots N_{(k)}$ over events represented by the rules $r \in \mathcal{R}_N$. For example, in EVG the nonterminal L^1_{NN} encodes three separate pieces of conditioning information: the direction $d = \mathit{left}$, the head part-of-speech $H = NN$, and the argument position $v = 0$; $\theta_{L^1_{NN} \to Y_{JJ}\ NN_L}$ represents the probability of generating JJ as the first left argument of NN. Suppose in EVG we are interested in smoothing $P(A \mid d, H, v)$ with a component that excludes the head conditioning event. Using linear interpolation, this would be:

$$P(A \mid d, H, v) = \lambda_1 P_1(A \mid d, H, v) + \lambda_2 P_2(A \mid d, v)$$

We will estimate PCFG rules with linearly interpolated probabilities by creating a tied PCFG which is extended by adding rules that select between the main distribution P_1 and the backoff distribution P_2, and also rules that correspond to draws from those distributions. We will make use of tied rule probabilities to make the independence assumption in the backoff distribution.

We still use the original grammar to parse the sentence. However, we estimate the parameters in the extended grammar and then translate them back into the original grammar for parsing.

More formally, suppose $\mathcal{B} \subseteq \mathcal{N}$ is a set of nonterminals (called the backoff set) with conditioning events $N_{(1)} \dots N_{(k-1)}$ in common (differing in a conditioning event $N_{(k)}$), and with rule sets of the same cardinality. If G is our model's PCFG, we can define a new tied PCFG $G' = (\mathcal{N}', \mathcal{T}, S, \mathcal{R}', \phi)$, where $\mathcal{N}' = \mathcal{N} \cup \left\{ N^{b_\ell} \mid N \in \mathcal{B},\ \ell \in \{1, 2\} \right\}$, meaning for each nonterminal N in the backoff set we add two nonterminals N^{b_1}, N^{b_2} representing each distribution P_1 and P_2. The new rule set $\mathcal{R}' = (\cup_{N \in \mathcal{N}'} \mathcal{R}'_N)$ where for all $N \in \mathcal{B}$ rule set $\mathcal{R}'_N = \left\{ N \to N^{b_\ell} \mid \ell \in \{1, 2\} \right\}$, meaning at N in G' we decide which distribution P_1, P_2 to use; and for $N \in \mathcal{B}$ and $\ell \in \{1, 2\}$, $\mathcal{R}'_{N^{b_\ell}} = \left\{ N^{b_\ell} \to \beta \mid N \to \beta \in \mathcal{R}_N \right\}$ indicating a draw from distribution P_ℓ. For nonterminals $N \notin \mathcal{B}$, $\mathcal{R}'_N = \mathcal{R}_N$. Finally, for each $N, M \in \mathcal{B}$ we specify a tying relation between the rules in $\mathcal{R}'_{N^{b_2}}$ and $\mathcal{R}'_{M^{b_2}}$, grouping together analogous rules. This has the effect of making an independence assumption about P_2, namely that it ignores the conditioning event $N_{(k)}$, drawing from a common distribution each time a nonterminal N^{b_2} is rewritten.

For example, in EVG to smooth $P(A = DT \mid d = \mathit{left}, H = NN, v = 0)$ with $P_2(A = DT \mid d = \mathit{left}, v = 0)$ we define the backoff set to be $\left\{ L^1_H \mid H \in V_\tau \right\}$. In the extended grammar we define the tying relation to form rule equivalence classes by the argument they generate, i.e. for each argument $A \in V_\tau$, we have a rule equivalence class $\left\{ L^{1b_2}_H \to Y_A\ H_L \mid H \in V_\tau \right\}$.

We can see that in grammar G' each $N \in \mathcal{B}$ eventually ends up rewriting to one of N's expansions β in G. There are two indirect paths, one through N^{b_1} and one through N^{b_2}. Thus this defines the probability of $N \to \beta$ in G, $\theta_{N \to \beta}$, as the probability of rewriting N as β in G' via N^{b_1} and N^{b_2}. That is:

$$\theta_{N \to \beta} = \phi_{N \to N^{b_1}} \phi_{N^{b_1} \to \beta} + \phi_{N \to N^{b_2}} \phi_{N^{b_2} \to \beta}$$

The example in Figure 6 shows the probability that L^1_{dog} rewrites to $Y_{big}\ dog_L$ in grammar G.

Typically when smoothing we need to incorporate the prior knowledge that conditioning events that have been seen fewer times should be more strongly smoothed. We accomplish this by setting the Dirichlet hyperparameters for each $N \to N^{b_1}, N \to N^{b_2}$ decision to $(K, 2K)$, where $K = |\mathcal{R}_{N^{b_1}}|$ is the number of rewrite rules for A. This ensures that the model will only start to ignore the backoff distribu-

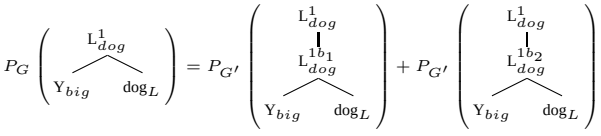


Figure 6: Using linear interpolation to smooth $L^1_{dog} \rightarrow Y_{big}\ dog_L$: The first component represents the distribution fully conditioned on head dog, while the second component represents the distribution ignoring the head conditioning event. This later is accomplished by tying the rule $L^{1b_2}_{dog} \rightarrow Y_{big}\ dog_L$ to, for instance, $L^{1b_2}_{cat} \rightarrow Y_{big}\ cat_L$, $L^{1b_2}_{fish} \rightarrow Y_{big}\ fish_L$ etc.

tion after having seen a sufficiently large number of training examples. [4]

4.1 Smoothed Dependency Models

Our first experiments examine smoothing the distributions over an argument in the DMV and EVG models. In DMV we smooth the probability of argument A given head part-of-speech H and direction d with a distribution that ignores H. In EVG, which conditions on H, d and argument position v we back off two ways. The first is to ignore v and use backoff conditioning event H, d. This yields a backoff distribution with the same conditioning information as the argument distribution from DMV. We call this EVG smoothed-skip-val.

The second possibility is to have the backoff distribution ignore the head part-of-speech H and use backoff conditioning event v, d. This assumes that arguments share a common distribution across heads. We call this EVG smoothed-skip-head. As we see below, backing off by ignoring the part-of-speech of the head H worked better than ignoring the argument position v.

For L-EVG we smooth the argument part-of-speech distribution (conditioned on the head word) with the unlexicalized EVG smoothed-skip-head model.

5 Initialization and Search issues

Klein and Manning (2004) strongly emphasize the importance of smart initialization in getting good performance from DMV. The likelihood function is full of local maxima and different initial parameter values yield vastly different quality solutions. They offer what they call a "harmonic initializer" which initializes the attachment probabilities to favor arguments that appear more closely in the data. This starts EM in a state preferring shorter attachments.

Since our goal is to expand the model to incorporate lexical information, we want an initialization scheme which does not depend on the details of DMV. The method we use is to create M sets of B random initial settings and to run VB some small number of iterations (40 in all our experiments) for each initial setting. For each of the M sets, the model with the best free energy of the B runs is then run out until convergence (as measured by likelihood of a held-out data set); the other models are pruned away. In this paper we use $B = 20$ and $M = 50$.

For the bth setting, we draw a random sample from the prior $\bar{\theta}^{(b)}$. We set the initial $Q(\mathbf{t}) = P(\mathbf{t}|\mathbf{s}, \bar{\theta}^{(b)})$ which can be calculated using the Expectation-Maximization E-Step. $Q(\bar{\theta})$ is then initialized using the standard VB M-step.

For the Lexicalized-EVG, we modify this procedure slightly, by first running MB smoothed EVG models for 40 iterations each and selecting the best model in each cohort as before; each L-EVG distribution is initialized from its corresponding EVG distribution. The new $P(A|h, H, d, v)$ distributions are set initially to their corresponding $P(A|H, d, v)$ values.

6 Results

We trained on the standard Penn Treebank WSJ corpus (Marcus et al., 1993). Following Klein and Manning (2002), sentences longer than 10 words after removing punctuation are ignored. We refer to this variant as WSJ10. Following Cohen et al. (2008), we train on sections 2-21, used 22 as a held-out development corpus, and present results evaluated on section 23. The models were all trained using Variational Bayes, and initialized as described in Section 5. To evaluate, we follow Cohen et al. (2008) in using the mean of the variational posterior Dirichlets as a point estimate $\bar{\theta}'$. For the unsmoothed models we decode by selecting the Viterbi parse given $\bar{\theta}'$, or $argmax_t P(t|s, \bar{\theta}')$.

For the smoothed models we find the Viterbi parse of the unsmoothed CFG, but use the smoothed probabilities. We evaluate against the gold standard

[4] We set the other Dirichlet hyperparameters to 1.

Model	Variant	Dir. Acc.
DMV	harmonic init	46.9*
DMV	random init	55.7 (8.0)
DMV	log normal-families	59.4*
DMV	shared log normal-families	62.4†
DMV	smoothed	61.2 (1.2)
EVG	random init	53.3 (7.1)
EVG	smoothed-skip-val	62.1 (1.9)
EVG	smoothed-skip-head	65.0 (5.7)
L-EVG	smoothed	**68.8** (4.5)

Table 1: Directed accuracy (DA) for WSJ10, section 23. *,† indicate results reported by Cohen et al. (2008), Cohen and Smith (2009) respectively. Standard deviations over 10 runs are given in parentheses

dependencies for section 23, which were extracted from the phrase structure trees using the standard rules by Yamada and Matsumoto (2003). We measure the percent accuracy of the directed dependency edges. For the lexicalized model, we replaced all words that were seen fewer than 100 times with "UNK." We ran each of our systems 10 times, and report the average directed accuracy achieved. The results are shown in Table 1. We compare to work by Cohen et al. (2008) and Cohen and Smith (2009).

Looking at Table 1, we can first of all see the benefit of randomized initialization over the harmonic initializer for DMV. We can also see a large gain by adding smoothing to DMV, topping even the logistic normal prior. The unsmoothed EVG actually performs worse than unsmoothed DMV, but both smoothed versions improve even on smoothed DMV. Adding lexical information (L-EVG) yields a moderate further improvement.

As the greatest improvement comes from moving to model EVG smoothed-skip-head, we show in Table 2 the most probable arguments for each val, dir, using the mean of the appropriate variational Dirichlet. For $d = right, v = 1$, $P(A|v, d)$ largely seems to acts as a way of grouping together various verb types, while for $d = left, v = 0$ the model finds that nouns tend to act as the closest left argument.

Dir,Val	Arg	Prob	Dir,Val	Arg	Prob
left, 0	NN	0.65	right, 0	NN	0.26
	NNP	0.18		RB	0.23
	DT	0.12		NNS	0.12
				IN	0.11
left, 1	CC	0.35	right, 1	IN	0.78
	RB	0.27			
	IN	0.18			

Table 2: Most likely arguments given valence and direction, according to smoothing distribution $P(arg|dir, val)$ in EVG smoothed-skip-head model with lowest free energy.

7 Conclusion

We present a smoothing technique for unsupervised PCFG estimation which allows us to explore more sophisticated dependency grammars. Our method combines linear interpolation with a Bayesian prior that ensures the backoff distribution receives probability mass. Estimating the smoothed model requires running the standard Variational Bayes on an extended PCFG. We used this technique to estimate a series of dependency grammars which extend DMV with additional valence and lexical information. We found that both were helpful in learning English dependency grammars. Our L-EVG model gives the best reported accuracy to date on the WSJ10 corpus.

Future work includes using lexical information more deeply in the model by conditioning argument words and valence on the lexical head. We suspect that successfully doing so will require using much larger datasets. We would also like to explore using our smoothing technique in other models such as HMMs. For instance, we could do unsupervised HMM part-of-speech induction by smooth a tritag model with a bitag model. Finally, we would like to learn the parts-of-speech in our dependency model from text and not rely on the gold-standard tags.

Acknowledgements

This research is based upon work supported by National Science Foundation grants 0544127 and 0631667 and DARPA GALE contract HR0011-06-2-0001. We thank members of BLLIP for their feedback.

References

Shay B. Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of NAACL-HLT 2009*.

Shay B. Cohen, Kevin Gimpel, and Noah A. Smith. 2008. Logistic normal priors for unsupervised probabilistic grammar induction. In *Advances in Neural Information Processing Systems 21*.

Michael Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, The University of Pennsylvania.

Jason Eisner and John Blatz. 2007. Program transformations for optimization of parsing algorithms and other weighted logic programs. In *Proceedings of the 11th Conference on Formal Grammar*.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *Proceedings of ACL 1999*.

Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of NAACL 2007*.

Mark Johnson. 2007. Transforming projective bilexical dependency grammars into efficiently-parsable CFGs with unfold-fold. In *Proceedings of ACL 2007*.

Dan Klein and Christopher Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of ACL 2002*.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL 2004*, July.

Kenichi Kurihara and Taisuke Sato. 2004. An application of the variational bayesian approach to probabilistics context-free grammars. In *IJCNLP 2004 Workshop Beyond Shallow Analyses*.

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

David McClosky. 2008. Modeling valence effects in unsupervised grammar induction. Technical Report CS-09-01, Brown University, Providence, RI, USA.

Noah A. Smith and Jason Eisner. 2005. Guiding unsupervised grammar induction using contrastive estimation. In *International Joint Conference on Artificial Intelligence Workshop on Grammatical Inference Applications*.

Noah A. Smith and Jason Eisner. 2006. Annealing structural bias in multilingual weighted grammar induction. In *Proceedings of COLING-ACL 2006*.

Noah A. Smith. 2006. *Novel Estimation Methods for Unsupervised Discovery of Latent Structure in Natural Language Text*. Ph.D. thesis, Department of Computer Science, Johns Hopkins University.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *In Proceedings of the International Workshop on Parsing Technologies*.

Context-Dependent Alignment Models for Statistical Machine Translation

Jamie Brunning, Adrià de Gispert and William Byrne
Machine Intelligence Laboratory
Department of Engineering, Cambridge University
Trumpington Street, Cambridge, CB2 1PZ, U.K.
{jjjb2,ad465,wjb31}@eng.cam.ac.uk

Abstract

We introduce alignment models for Machine Translation that take into account the context of a source word when determining its translation. Since the use of these contexts alone causes data sparsity problems, we develop a decision tree algorithm for clustering the contexts based on optimisation of the EM auxiliary function. We show that our context-dependent models lead to an improvement in alignment quality, and an increase in translation quality when the alignments are used in Arabic-English and Chinese-English translation.

1 Introduction

Alignment modelling for Statistical Machine Translation (SMT) is the task of determining translational correspondences between the words in pairs of sentences in parallel text. Given a source language word sequence f_1^J and a target language word sequence e_1^I, we model the translation probability as $\mathrm{P}(e_1^I|f_1^J)$ and introduce a hidden variable a_1^I representing a mapping from the target word positions to source word positions such that e_i is aligned to f_{a_i}. Then $\mathrm{P}(e_1^I|f_1^j) = \sum_{a_1^I} \mathrm{P}(e_1^I, a_1^I|f_1^j)$ (Brown et al., 1993).

Previous work on statistical alignment modelling has not taken into account the source word context when determining translations of that word. It is intuitive that a word in one context, with a particular part-of-speech and particular words surrounding it, may translate differently when in a different context. We aim to take advantage of this information to provide a better estimate of the word's translation. The challenge of incorporating context information is maintaining computational tractability of estimation and alignment, and we develop algorithms to overcome this.

The development of efficient estimation procedures for context-dependent acoustic models revolutionised the field of Automatic Speech Recognition (ASR) (Young et al., 1994). Clustering is used extensively for improving parameter estimation of triphone (and higher order) acoustic models, enabling robust estimation of parameters and reducing the computation required for recognition. Kannan et al. (1994) introduce a binary tree-growing procedure for clustering Gaussian models for triphone contexts based on the value of a likelihood ratio. We adopt a similar approach to estimate context-dependent translation probabilities.

We focus on alignment with IBM Model 1 and HMMs. HMMs are commonly used to generate alignments from which state of the art SMT systems are built. Model 1 is used as an intermediate step in the creation of more powerful alignment models, such as HMMs and further IBM models. In addition, it is used in SMT as a feature in Minimum Error Training (Och et al., 2004) and for rescoring lattices of translation hypotheses (Blackwood et al., 2008). It is also used for lexically-weighted phrase extraction (Costa-jussà and Fonollosa, 2005) and sentence segmentation of parallel text (Deng et al., 2007) prior to machine translation.

1.1 Overview

We first develop an extension to Model 1 that allows the use of arbitrary context information about a source word to estimate context-dependent word-to-word translation probabilities. Since there is insufficient training data to accurately estimate translation probabilities for less frequently occurring contexts, we develop a decision tree clustering algorithm to form context classes. We go on to develop a context-dependent HMM model for alignment.

In Section 3, we evaluate our context-dependent models on Arabic-English parallel text, comparing them to our baseline context-independent models. We perform morphological decomposition of the Arabic text using MADA, and use part-of-speech taggers on both languages. Alignment quality is measured using Alignment Error Rate (AER) measured against a manually-aligned parallel text. Section 4 uses alignments produced by

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 110–118,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

our improved alignment models to initialise a statistical machine translation system and evaluate the quality of translation on several data sets. We also apply part-of-speech tagging and decision tree clustering of contexts to Chinese-English parallel text; translation results for these languages are presented in Section 4.2.

1.2 Previous and related work

Brown et al. (1993) introduce IBM Models 1-5 for alignment modelling; Vogel et al. (1996) propose a Hidden Markov Model (HMM) model for word-to-word alignment, where the words of the source sentence are viewed as states of an HMM and emit target sentence words; Deng and Byrne (2005a) extend this to an HMM word-to-phrase model which allows many-to-one alignments and can capture dependencies within target phrases.

Habash and Sadat (2006) perform morphological decomposition of Arabic words, such as splitting of prefixes and suffixes. This leads to gains in machine translation quality when systems are trained on parallel text containing the modified Arabic and processing of Arabic text is carried out prior to translation. Nießen and Ney (2001a) perform pre-processing of German and English text before translation; Nießen and Ney (2001b) use morphological information of the current word to estimate hierarchical translation probabilities.

Berger et al. (1996) introduce maximum entropy models for machine translation, and use a window either side of the target word as context information. Varea et al. (2002) test for the presence of specific words within a window of the current source word to form features for use inside a maximum entropy model of alignment.

Toutanova et al. (2002) use part-of-speech information in both the source and target languages to estimate alignment probabilities, but this information is not incorporated into translation probabilities. Popović and Ney (2004) use the base form of a word and its part-of-speech tag during the estimation of word-to-word translation probabilities for IBM models and HMMs, but do not defined context-dependent estimates of translation probabilities.

Stroppa et al. (2007) consider context-informed features of phrases as components of the log-linear model during phrase-based translation, but do not address alignment.

2 Use of source language context in alignment modelling

Consider the alignment of the target sentence $\mathbf{e} = e_1^I$ with the source sentence $\mathbf{f} = f_1^J$. Let $\mathbf{a} = a_1^I$ be the alignments of the target words to the source words. Let c_j be the context information of f_j for $j = 1, \ldots, J$. This context information can be any information about the word, e.g. part-of-speech, previous and next words, part-of-speech of previous and next words, or longer range context information.

We follow Brown et al. (1993), but extend their modelling framework to include information about the source word from which a target word is emitted. We model the alignment process as:

$$\begin{aligned}\mathrm{P}(e_1^I, a_1^I, I|f_1^J, c_1^J) = \\ \mathrm{P}(I|f_1^J, c_1^J) \prod_{i=1}^{I} \big[\mathrm{P}(e_i|a_1^i, e_1^{i-1}, f_1^J, c_1^J, I) \\ \times \mathrm{P}(a_i|e_1^{i-1}, a_1^{i-1}, f_1^J, c_1^J, I)\big] \quad (1)\end{aligned}$$

We introduce word-to-word translation tables that depend on the source language context for each word, i.e. the probability that f translates to e given f has context c is $t(e|f, c)$. We assume that the context sequence is given for a source word sequence. This assumption can be relaxed to allow for multiple tag sequences as hidden processes, but we assume here that a tagger generates a single context sequence c_1^J for a word sequence f_1^J. This corresponds to the assumption that, for a context sequence $\tilde{c}_1^J$, $\mathrm{P}(\tilde{c}_1^J|f_1^J) = \delta_{c_1^J}(\tilde{c}_1^J)$; hence

$$\mathrm{P}(e_1^I, a_1^I|f_1^J) = \sum_{\tilde{c}_1^J} \mathrm{P}(e_1^I, a_1^I, \tilde{c}_1^J|f_1^J) = \mathrm{P}(e_1^I, a_1^I|c_1^J, f_1^J)$$

For Model 1, ignoring the sentence length distribution,

$$\mathrm{P}_{\mathrm{M1}}(e_1^I, a_1^I|f_1^J, c_1^J) = \frac{1}{(J+1)^I} \prod_{i=1}^{I} t(e_i|f_{a_i}, c_{a_i}). \quad (2)$$

Estimating translation probabilities separately for every possible context of a source word individually leads to problems with data sparsity and rapid growth of the translation table. We therefore wish to cluster source contexts which lead to similar probability distributions. Let C_f denote the set of all observed contexts of source word f. A particular clustering is denoted

$$\mathcal{K}_f = \{K_{f,1}, \ldots, K_{f,N_f}\},$$

where $\mathcal{K}_f$ is a partition of C_f. We define a class membership function μ_f such that for any context c, $\mu_f(c)$ is the cluster containing c. We assume that all contexts in a cluster give rise to the same translation probability distribution for that source word, i.e. for a cluster K, $t(e|f, c) = t(e|f, c')$ for all contexts $c, c' \in K$ and all target words e; we write this shared translation probability as $t(e|f, K)$.

The Model 1 sentence translation probability for a given alignment (Equation 2) becomes

$$\mathrm{P}_{\mathrm{M1}}(e_1^I, a_1^I|f_1^J, c_1^J) = \frac{1}{(J+1)^I} \prod_{i=1}^{I} t(e_i|f_{a_i}, \mu_f(c_{a_i})). \quad (3)$$

For HMM alignment, we assume that the transition probabilities $a(a_i|a_{i-1})$ are independent of the word contexts and the sentence translation probability is

$$\mathrm{P_H}(e_1^I, a_1^I|f_1^J, c_1^J) = \prod_{i=1}^{I} a(a_i|a_{i-1}, J)t(e_i|f_{a_i}, \mu_f(c_{a_i})). \quad (4)$$

Section 2.1.1 describes how the context classes are determined by optimisation of the EM auxiliary function. Although the translation model is significantly more complex than that of context-independent models, once class membership is fixed, alignment and parameter estimation use the standard algorithms.

2.1 EM parameter estimation

We train using Expectation Maximisation (EM), optimising the log probability of the training set $\{\mathbf{e}^{(s)}, \mathbf{f}^{(s)}\}_{s=1}^{S}$ (Brown et al., 1993). Given model parameters θ', we estimate new parameters θ by maximisation of the EM auxiliary function

$$\sum_{s,\mathbf{a}} \mathrm{P}_{\theta'}(\mathbf{a}|\mathbf{f}^{(s)}, \mathbf{c}^{(s)}, \mathbf{e}^{(s)}) \log \mathrm{P}_{\theta}(\mathbf{e}^{(s)}, \mathbf{a}, I^{(s)}|\mathbf{f}^{(s)}, \mathbf{c}^{(s)}).$$

We assume the sentence length distribution and alignment probabilities do not depend on the contexts of the source words; hence the relevant part of the auxiliary function is

$$\sum_e \sum_f \sum_{c \in C_f} \gamma'(e|f,c) \log t(e|f,c), \quad (5)$$

where

$$\gamma'(e|f,c) = \sum_s \sum_{i=1}^{I^{(s)}} \sum_{j=1}^{J^{(s)}} \Big[\delta_c(c_j^{(s)})\delta_e(e_i^{(s)})\delta_f(f_j^{(s)}) \times \mathrm{P}_{\theta'}(a_i = j|\mathbf{e}^{(s)}, \mathbf{f}^{(s)}, \mathbf{c}^{(s)})\Big]$$

Here γ' can be computed under Model 1 or the HMM, and is calculated using the forward-backward algorithm for the HMM.

2.1.1 Parameter estimation with clustered contexts

We can re-write the EM auxiliary function (Equation 5) in terms of the cluster-specific translation probabilities:

$$\begin{aligned} &\sum_e \sum_f \sum_{l=1}^{|\mathcal{K}_f|} \sum_{c \in K_{f,l}} \gamma'(e|f,c) \log t(e|f,c) \\ &= \sum_e \sum_f \sum_{K \in \mathcal{K}_f} \gamma'(e|f,K) \log t(e|f,K) \quad (6) \\ &\text{where } \gamma'(e|f,K) = \sum_{c \in K} \gamma'(e|f,c) \end{aligned}$$

Following the usual derivation, the EM update for the class-specific translation probabilities becomes

$$\hat{t}(e|f,K) = \frac{\gamma'(e|f,K)}{\sum_{e'} \gamma'(e'|f,K)}. \quad (7)$$

Standard EM training can be viewed a special case of this, with every context of a source word grouped into a single cluster. Another way to view these clustered context-dependent models is that contexts belonging to the same cluster are tied and share a common translation probability distribution, which is estimated from all training examples in which any of the contexts occur.

2.2 Decision trees for context clustering

The objective for each source word is to split the contexts into classes to maximise the likelihood of the training data. Since it is not feasible to maximise the likelihood of the observations directly, we maximise the expected log likelihood by considering the EM auxiliary function, in a similar manner to that used for modelling contextual variations of phones for ASR (Young et al., 1994; Singer and Ostendorf, 1996). We perform divisive clustering independently for each source word f, by building a binary decision tree which forms classes of contexts which maximise the EM auxiliary function. Questions for the tree are drawn from a set of questions $\mathcal{Q} = \{q_1, \ldots, q_{|\mathcal{Q}|}\}$ concerning the context information of f.

Let K be any set of contexts of f, and define

$$\begin{aligned} L(K) &= \sum_e \sum_{c \in K} \gamma'(e|f,c) \log t(e|f,c) \\ &= \sum_e \sum_{c \in K} \gamma'(e|f,c) \log \frac{\sum_{c \in K} \gamma'(e|f,c)}{\sum_{e'} \sum_{c \in K} \gamma'(e'|f,c)}. \end{aligned}$$

This is the contribution to the EM auxiliary function of source word f occurring in the contexts of K. Let q be a binary question about the context of f, and consider the effect on the partial auxiliary function (Equation 6) of splitting K into two clusters using question q. Define K_q be the set of contexts in K which answer 'yes' to q and $K_{\bar{q}}$ be the contexts which answer 'no'. Define the objective function

$$\begin{aligned} Q_{f,q}(K) &= \sum_e \sum_{c \in K_q} \gamma'(e|f,c) \log t(e|f,c) \\ &\quad + \sum_e \sum_{c \in K_{\bar{q}}} \gamma'(e|f,c) \log t(e|f,c) \\ &= L(K_q) + L(K_{\bar{q}}) \end{aligned}$$

When the node is split using question q, the increase in objective function is given by

$$Q_{f,q}(K) - L(K) = L(K_{\bar{q}}) + L(K_q) - L(K).$$

We choose q to maximise this.

In order to build the decision tree for f, we take the set of all contexts C_f as the initial cluster at the root node. We then find the question $\hat{q}$ such that $Q_{f,q}(C_f)$ is maximal, i.e.

$$\hat{q} = \arg\max_{q \in \mathcal{Q}} Q_{f,q}(C_f)$$

This splits C_f, so our decision tree now has two nodes. We iterate this process, at each iteration splitting (into two further nodes) the leaf node that leads to the greatest increase in objective function. This leads to a greedy search to optimise the log likelihood over possible state clusterings.

In order to control the growth of the tree, we put in place two thresholds:

- T_{imp} is the minimum improvement in objective function required for a node to be split; without it, we would continue splitting nodes until each contained only one context, even though doing so would cause data sparsity problems.
- T_{occ} is the minimum occupancy of a node, based on how often the contexts at that node occur in the training data; we want to ensure that there are enough examples of a context in the training data to estimate accurately the translation probability distribution for that cluster.

For each leaf node l and set of contexts K_l at that node, we find the question q_l that, when used to split K_l, produces the largest gain in objective function:

$$\begin{aligned} q_l &= \arg\max_{q \in \mathcal{Q}} [L(K_{l,q}) + L(K_{l,\bar{q}}) - L(K_l)] \\ &= \arg\max_{q \in \mathcal{Q}} [L(K_{l,q}) + L(K_{l,\bar{q}})] \end{aligned}$$

We then find the leaf node for which splitting gives the largest improvement:

$$\hat{l} = \arg\max_{l} [L(K_{l,q_l}) + L(K_{l,\bar{q}_l}) - L(K_l)]$$

If the following criteria are both satisfied at that node, we split the node into two parts, creating two leaf nodes in its place:

- The objective function increases sufficiently

$$L(K_{l,q_l}) + L(K_{l,\bar{q}_l}) - L(K_l) > T_{\text{imp}}$$

- The occupancy threshold is exceeded for both child nodes:

$$\sum_e \sum_{c \in K_{l,x}} \gamma'(e|f,c) > T_{\text{occ}} \text{ for } x = q, \bar{q}$$

We perform such clustering for every source word in the parallel text.

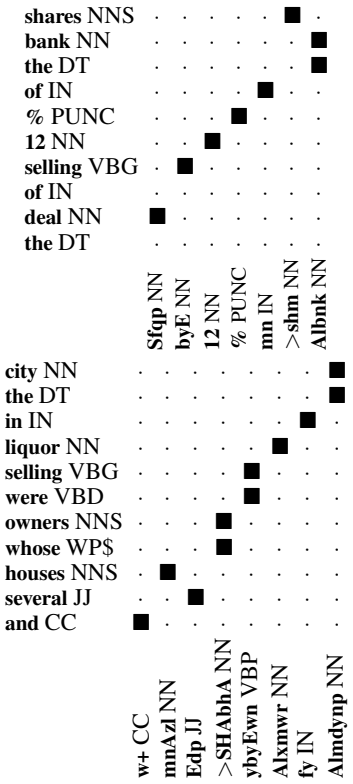


Figure 1: Alignment of the English *selling* in different contexts. In the first, it is preceded by *of* and links to the infinitive of the Arabic verb *byE*; in the second, it is preceded by *were* and links to an inflected form of the same Arabic verb, *ybyEwn*.

3 Evaluation of alignment quality

Our models were built using the MTTK toolkit (Deng and Byrne, 2005b). Decision tree clustering was implemented and the process parallelised to enable thousands of decision trees to be built. Our context-dependent (CD) Model 1 models trained on context-annotated data were compared to the baseline context-independent (CI) models trained on untagged data.

The models were trained using data allowed for the NIST 08 Arabic-English evaluation[1], excluding the UN collections, comprising 300k parallel sentence pairs, a total of 8.4M words of Arabic and 9.5M words of English.

The Arabic language incorporates into its words several prefixes and suffixes which determine grammatical features such as gender, number, person and voice. The MADA toolkit (Habash and Sadat, 2006) was used to perform Arabic morphological word decomposition and part-of-speech tagging. It determines the best analysis for each word in a sentence and splits word prefixes and suffixes, based on the alternative analyses provided by BAMA (Buckwalter, 2002). We use tokenisation scheme

[1] http://nist.gov/speech/tests/mt/2008

'D2', which splits certain prefixes and has been reported to improve machine translation performance (Habash and Sadat, 2006). The alignment models are trained on this processed data, and the prefixes and suffixes are treated as words in their own right; in particular their contexts are examined and clustered.

The TnT tagger (Brants, 2000), used as distributed with its model trained on the Wall Street Journal portion of the Penn treebank, was used to obtain part-of-speech tags for the English side of the parallel text. Marcus et al. (1993) gives a complete list of part-of-speech tags produced. No morphological analysis is performed for English.

Automatic word alignments were compared to a manually-aligned corpus made up of the IBM Arabic-English Word Alignment Corpus (Ittycheriah et al., 2006) and the word alignment corpora LDC2006E86 and LDC2006E93. This contains 28k parallel text sentences pairs: 724k words of Arabic and 847k words of English. The alignment links were modified to reflect the MADA tokenisation; after modification, there are 946k word-to-word alignment links.

Alignment quality was evaluated by computing Alignment Error Rate (AER) (Och and Ney, 2000) relative to the manual alignments. Since the links supplied contain only 'sure' links and no 'possible' links, we use the following formula for computing AER given reference alignment links S and hypothesised alignment links A: $\text{AER} = 1 - \frac{2|S \cap A|}{|S|+|A|}$.

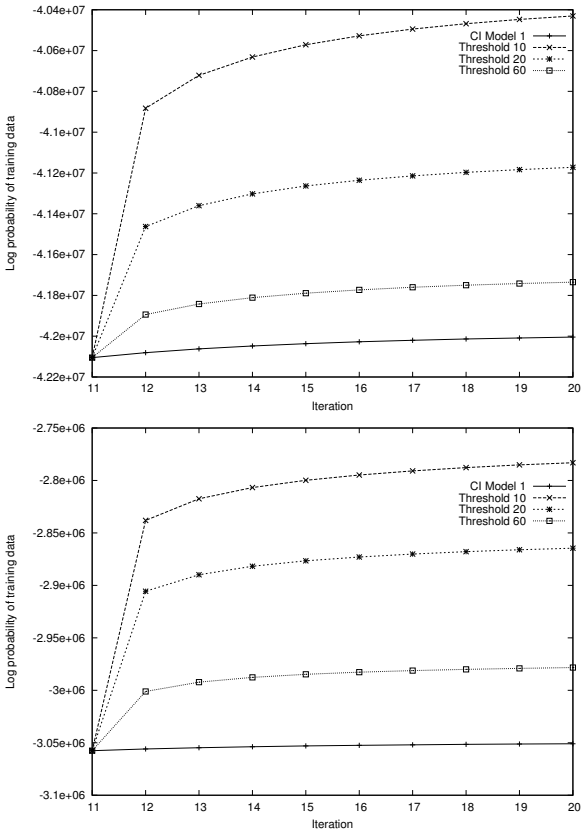


Figure 2: Increase in log probability of training data during training for varying T_{imp}, with Model 1, for Arabic to English (top) and English to Arabic (bottom)

3.1 Questions about contexts

The algorithm presented in Section 2 allows for any information about the context of the source word to be considered. We could consider general questions of the form *'Is the previous word x?'* and *'Does word y occur within n words of this one?'*. To maintain computational tractability, we restrict the questions to those concerning the part-of-speech tag assigned to the current, previous and next words. We do not ask questions about the identities of the words themselves. For each part-of-speech tag T, we ask the question *'Does w have tag T?'*. In addition, we group part-of-speech tags to ask more general questions: e.g. the set of contexts which satisfies *'Is w a noun?'* contains those that satisfy *'Is w a proper noun?'* and *'Is w a singular or mass noun?'*. We also ask the same questions of the previous and next words in the source sentence. In English, this gives a total of 152 distinct questions, each of which is considered when splitting a leaf node. The MADA part-of-speech tagger uses a reduced tag set, which produces a total of 68 distinct questions.

Figure 1 shows the links of the English source word *selling* in two different contexts where it links to different words in Arabic, which are both forms of the same verb. The part-of-speech of the previous word is useful for discriminating between the two cases, whereas a context-independent model would assign the same probability to both Arabic words.

3.2 Training Model 1

Training is carried out in both translation directions. For Arabic to English, the Arabic side of the parallel text is tagged and the English side remains untagged; we view the English words as being generated from the Arabic words and questions are asked about the context of the Arabic words to determine clusters for the translation table. For English to Arabic, the situation is reversed: we used tagged English text as the source language and untagged Arabic text, with morphological decomposition, as the target language.

Standard CI Model 1 training, initialised with a uniform translation table so that $t(e|f)$ is constant for all source/target word pairs (f, e), was run on untagged data for 10 iterations in each direction (Brown et al., 1993; Deng and Byrne, 2005b). A decision tree was built to cluster the contexts and a further 10 iterations of training were carried out using the tagged words-with-context to produce context-dependent models (CD Model 1). The

English question	Frequency
Is_Next_Preposition	1523
Is_Prev_Determiner	1444
Is_Prev_Preposition	1209
Is_Prev_Adjective	864
Is_Next_Noun_Singular_Mass	772
Is_Prev_Noun_Singular_Mass	690
Is_Next_Noun_Plural	597
Is_Next_Noun	549
Arabic question	**Frequency**
Is_Prev_Preposition	1110
Is_Next_Preposition	993
Is_Prev_Noun	981
Is_Next_Noun	912
Is_Prev_Coordinating_Conjunction	627
Is_Prev_Noun_SingularMass	607
Is_Next_Punctuation	603
Is_Next_Adjective_Adverb	559

Table 1: Most frequent root node context questions

models were then evaluated using AER at each training iteration. A number of improvement thresholds T_{imp} were tested, and performance compared to that of models found after further iterations of CI Model 1 training on the untagged data. In both alignment directions, the log probability of the training data increases during training (see Figure 2). As expected, the training set likelihood increases as the threshold T_{imp} is reduced, allowing more clusters and closer fitting to the data.

3.2.1 Analysis of frequently used questions

Table 1 shows the questions used most frequently at the root node of the decision tree when clustering contexts in English and Arabic. Because they are used first, these are the questions that individually give the greatest ability to discriminate between the different contexts of a word. The list shows the importance of the left and right contexts of the word in predicting its translation: of the most common 50 questions, 25 concern the previous word, 19 concern the next, and only 6 concern the part-of-speech of the current word. For Arabic, of the most frequent 50 questions, 21 concern the previous word, 20 concern the next and 9 the current word.

3.2.2 Alignment Error Rate

Since MT systems are usually built on the union of the two sets of alignments (Koehn et al., 2003), we consider the union of alignments in the two directions as well as those in each direction. Figure 3 shows the change in AER of the alignments in each direction, as well as the alignment formed by taking their union at corresponding thresholds and training iterations.

T_{imp}	Arabic-English (%)	English-Arabic (%)
10	30601 (25.33)	26011 (39.87)
20	11193 (9.27)	18365 (28.15)
40	1874 (1.55)	9104 (13.96)
100	307 (0.25)	1128 (1.73)

Table 2: Words [number (percentage)] with context-dependent translation for varying T_{imp}

3.2.3 Variation of improvement threshold T_{imp}

There is a trade-off between modelling the data accurately, which requires more clusters, and eliminating data sparsity problems, which requires each cluster to contain contexts that occur frequently enough in the training data to estimate the translation probabilities accurately. Use of a smaller threshold T_{imp} leads to more clusters per word and an improved ability to fit to the data, but this can lead to reduced alignment quality if there is insufficient data to estimate the translation probability distribution accurately for each cluster. For lower thresholds, we observe over-fitting and the AER rises after the second iteration of CD training, similar to the behaviour seen in Och (2002). Setting $T_{\text{imp}} = 0$ results in each context of a word having its own cluster, which leads to data sparsity problems.

Table 2 shows the percentage of words for which the contexts are split into multiple clusters for CD Model 1 with varying improvement thresholds. This occurs when there are enough training data examples and sufficient variability between the contexts of a word that splitting the contexts into more than one cluster increases the EM auxiliary function. For words where the contexts are not split, all the contexts remain in the same cluster and parameter estimation is exactly the same as for the unclustered context-independent models.

3.3 Training HMMs

Adding source word context to translation has so far led to improvements in AER for Model 1, but the performance does not match that of HMMs trained on untagged data; we therefore train HMMs on tagged data.

We proceed with Model 1 and Model 2 trained in the usual way, and context-independent (CI) HMMs were trained for 5 iterations on the untagged data. Statistics were then gathered for clustering at various thresholds, after which 5 further EM iterations were performed with tagged data to produce context-dependent (CD) HMMs. The HMMs were trained in both the Arabic to English and the English to Arabic directions. The log likelihood of the training set varies with T_{imp} in much the same way as for Model 1, increasing at each iteration, with greater likelihood at lower thresholds. Figure 4 shows how the AER of the union alignment varies with T_{imp} during training. As with Model 1, the clustered HMM

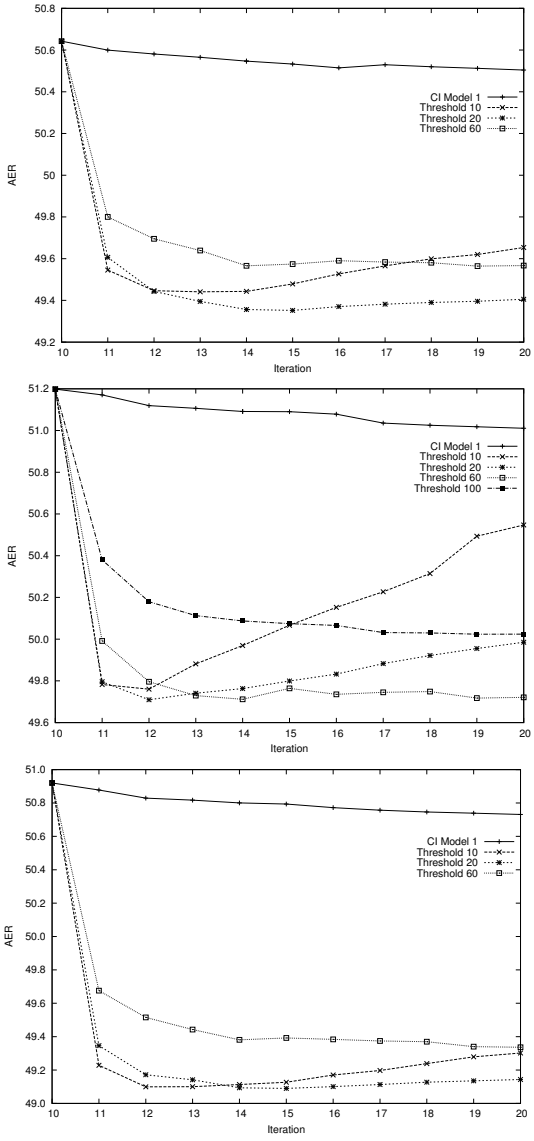


Figure 3: Variation of AER during Model 1 training for varying T_{imp}, for Arabic to English (top), English to Arabic (middle) and their union (bottom)

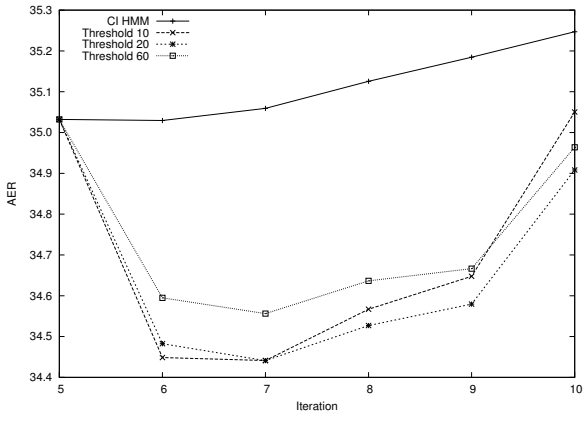


Figure 4: AER of the union alignment for varying T_{imp} with the HMM model

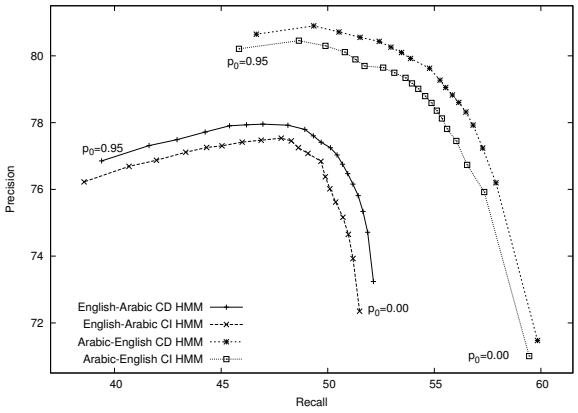


Figure 5: Precision/recall curves for the context-dependent HMM and the baseline context-independent HMM, for Arabic to English and English to Arabic. p_0 varies from 0.00 to 0.95 in steps of 0.05.

models produce alignments with a lower AER than the baseline model, and there is evidence of over-fitting to the training data.

3.3.1 Alignment precision and recall

The HMM models include a *null transition probability*, p_0, which can be modified to adjust the number of alignments to the null token (Deng and Byrne, 2005a). Where a target word is emitted from null, it is not included in the alignment links, so this target word is viewed as not being aligned to any source word; this affects the precision and recall. The results reported above use $p_0 = 0.2$ for English-Arabic and $p_0 = 0.4$ for Arabic-English; we can tune these values to produce alignments with the lowest AER. Figure 5 shows precision-recall curves for the CD HMMs compared to the CI HMMs for both translation directions. For a given value of precision, the CD HMM has higher recall; for a given value of recall, the CD HMM has higher precision.

We do not report F-score (Fraser and Marcu, 2006) since in our experiments we have not found strong correlation with translation performance, but we note that these results for precision and recall should lead to improved F-scores as well.

4 Evaluation of translation quality

We have shown that the context-dependent models produce a decrease in AER measured on manually-aligned data; we wish to show this improved model performance leads to an increase in translation quality, measured by BLEU score (Papineni et al., 2001). In addition to the Arabic systems already evaluated by AER, we also report results for a Chinese-English translation system.

Alignment models were evaluated by aligning the training data using the models in each translation direc-

tion. HiFST, a WFST-based hierarchical translation system described in (Iglesias et al., 2009), was trained on the union of these alignments. MET (Och, 2003) was carried out using a development set, and the BLEU score evaluated on two test sets. Decoding used a 4-gram language model estimated from the English side of the entire MT08 parallel text, and a 965M word subset of monolingual data from the English Gigaword Third Edition.

For both Arabic and English, the CD HMM models were evaluated as follows. Iteration 5 of the CI HMM was used to produce alignments for the parallel text training data: these were used to train the baseline system. The same data is aligned using CD HMMs after two further iterations of training and a second WFST-based translation system built from these alignments. The models are evaluated by comparing BLEU scores with those of the baseline model.

4.1 Arabic to English translation

Alignment models were trained on the NIST MT08 Arabic-English parallel text, excluding the UN portion. The null alignment probability was chosen based on the AER, resulting in values of $p_0 = 0.05$ for Arabic to English and $p_0 = 0.10$ for English to Arabic. We perform experiments on the NIST Arabic-English translation task. The *mt02_05_tune* and *mt02_05_test* data sets are formed from the odd and even numbered sentences of the NIST MT02 to MT05 evaluation sets respectively; each contains 2k sentences and 60k words. We use *mt02_05_tune* as a development set and evaluate the system on *mt02_05_test* and the newswire portion of the MT08 set, *MT08-nw*. Table 3 shows a comparison of the system trained using CD HMMs with the baseline system, which was trained using CI HMM models on untagged data. The context-dependent models result in a gain in BLEU score of 0.3 for *mt02_05_test* and 0.6 for *MT08-nw*.

4.2 Chinese to English translation

The Chinese training set was 600k random parallel text sentences of the newswire LDC collection allowed for NIST MT08, a total of 15.2M words of Chinese and 16.6M words of English. The Chinese text was tagged using the MXPOST maximum-entropy part of speech tagging tool (Ratnaparkhi, 1996) trained on the Penn Chinese Treebank 5.1; the English text was tagged using the TnT part of speech tagger (Brants, 2000) trained on the Wall Street Journal portion of the English Penn treebank.

The development set *tune-nw* and validation set *test-nw* contain a mix of the newswire portions of MT02 through MT05 and additional developments sets created by translation within the GALE program. We also report results on the newswire portion of the MT08 set. Again we see an increase in BLEU score for both test sets: 0.5 for *test-nw* and 0.8 for *MT08-nw*.

Arabic-English			
Alignments	tune	mt02_05_test	MT08-nw
CI HMM	50.0	49.4	46.3
CD HMM	50.0	49.7	46.9
Chinese-English			
Alignments	tune	test-nw	MT08-nw
CI HMM	28.1	28.5	26.9
CD HMM	28.5	29.0	27.7

Table 3: Comparison, using BLEU score, of the CD HMM with the baseline CI HMM

5 Conclusions and future work

We have introduced context-dependent Model 1 and HMM alignment models, which use context information in the source language to improve estimates of word-to-word translation probabilities. Estimation of parameters using these contexts without smoothing leads to data sparsity problems; therefore we have developed decision tree clustering algorithms to cluster source word contexts based on optimisation of the EM auxiliary function. Context information is incorporated by the use of part-of-speech tags in both languages of the parallel text, and the EM algorithm is used for parameter estimation.

We have shown that these improvements to the model lead to decreased AER compared to context-independent models. Finally, we compare machine translation systems built using our context-dependent alignments. For both Arabic- and Chinese-to-English translation, we report an increase in translation quality measured by BLEU score compared to a system built using context-independent alignments.

This paper describes an initial investigation into context-sensitive alignment models, and there are many possible directions for future research. Clustering the probability distributions of infrequently occurring may produce improvements in alignment quality, different model training schemes and extensions of the context-dependence to more sophisticated alignment models will be investigated. Further translation experiments will be carried out.

Acknowledgements

This work was supported in part by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022. J. Brunning is supported by a Schiff Foundation graduate studentship. Thanks to Yanjun Ma, Dublin City University, for training the Chinese part of speech tagger.

References

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

Graeme Blackwood, Adrià de Gispert, Jamie Brunning, and William Byrne. 2008. European language translation with weighted finite state transducers: The CUED MT system for the 2008 ACL workshop on SMT. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 131–134, Columbus, Ohio, June. Association for Computational Linguistics.

Thorsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference: ANLP-2000*, Seattle, USA.

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

T. Buckwalter. 2002. Buckwalter Arabic morphological analyzer.

Marta Ruiz Costa-jussà and Jos´e A. R. Fonollosa. 2005. Improving phrase-based statistical translation by modifying phrase extraction and including several features. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 149–154, June.

Yonggang Deng and William Byrne. 2005a. HMM word and phrase alignment for statistical machine translation. In *Proc. of HLT-EMNLP*.

Yonggang Deng and William Byrne. 2005b. JHU-Cambridge statistical machine translation toolkit (MTTK) user manual.

Yonggang Deng, Shankhar Kumar, and William Byrne. 2007. Segmentation and alignment of parallel text for statistical machine translation. *Journal of Natural Language Engineering*, 13:3:235–260.

Alexander Fraser and Daniel Marcu. 2006. Measuring word alignment quality for statistical machine translation. Technical Report ISI-TR-616, ISI/University of Southern California, May.

Nizar Habash and Fatiha Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *HLT-NAACL*.

G. Iglesias, A. de Gispert, E. R. Banga, and W. Byrne. 2009. Hierarchical phrase-based translation with weighted finite state transducers. In *Procedings of NAACL-HLT, 2009*, Boulder, Colorado.

Abraham Ittycheriah, Yaser Al-Onaizan, and Salim Roukos. 2006. The IBM Arabic-English word alignment corpus, August.

A. Kannan, M. Ostendorf, and J. R. Rohlicek. 1994. Maximum likelihood clustering of Gaussians for speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 2(3):453–455, July.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54.

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

Sonja Nießen and Hermann Ney. 2001a. Morpho-syntactic analysis for reordering in statistical machine translation. In *Proceedings of MT Summit VIII*, pages 247–252, September.

Sonja Nießen and Hermann Ney. 2001b. Toward hierarchical models for statistical machine translation of inflected languages. In *Proceedings of the workshop on Data-driven methods in machine translation*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2000. A comparison of alignment models for statistical machine translation. In *Proceedings of the 18th conference on Computational Linguistics*, pages 1086–1090.

F. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of NAACL*.

Franz Josef Och. 2002. *Statistical Machine Translation: From Single Word Models to Alignment Templates*. Ph.D. thesis, Franz Josef Och.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Maja Popović and Hermann Ney. 2004. Improving word alignment quality using morpho-syntactic information. In *In Proceedings of COLING*, page 310.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142.

H. Singer and M. Ostendorf. 1996. Maximum likelihood successive state splitting. *Proceedings of ICASSP*, 2:601–604.

Nicolas Stroppa, Antal van den Bosch, and Andy Way. 2007. Exploiting source similarity for SMT using context-informed features. In *Proceedings of the 11th Conference on Theoretical and Methodological Issues in Machine Translation (TMI 2007)*, pages 231 – 240.

Kristina Toutanova, H. Tolga Ilhan, and Christopher D. Manning. 2002. Extensions to HMM-based statistical word alignment models. In *Proceedings of EMNLP*, pages 87–94.

Ismael García Varea, Franz J. Och, Hermann Ney, and Francisco Casacuberta. 2002. Improving alignment quality in statistical machine translation using context-dependent maximum entropy models. In *Proceedings of COLING*, pages 1–7.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING*, pages 836–841.

S. J. Young, J. J. Odell, and P. C. Woodland. 1994. Tree-based state tying for high accuracy acoustic modelling. In *HLT '94: Proceedings of the workshop on Human Language Technology*, pages 307–312.

Graph-based Learning for Statistical Machine Translation

Andrei Alexandrescu
Dept. of Comp. Sci. Eng.
University of Washington
Seattle, WA 98195, USA
andrei@cs.washington.edu

Katrin Kirchhoff
Dept. of Electrical Eng.
University of Washington
Seattle, WA 98195, USA
katrin@ee.washington.edu

Abstract

Current phrase-based statistical machine translation systems process each test sentence in isolation and do not enforce global consistency constraints, even though the test data is often internally consistent with respect to topic or style. We propose a new consistency model for machine translation in the form of a graph-based semi-supervised learning algorithm that exploits similarities between training and test data and also similarities between different test sentences. The algorithm learns a regression function jointly over training and test data and uses the resulting scores to rerank translation hypotheses. Evaluation on two travel expression translation tasks demonstrates improvements of up to 2.6 BLEU points absolute and 2.8% in PER.

1 Introduction

Current phrase-based statistical machine translation (SMT) systems commonly operate at the sentence level—each sentence is translated in isolation, even when the test data consists of internally coherent paragraphs or stories, such as news articles. For each sentence, SMT systems choose the translation hypothesis that maximizes a combined log-linear model score, which is computed independently of all other sentences, using globally optimized combination weights. Thus, similar input strings may be translated in very different ways, depending on which component model happens to dominate the combined score for that sentence. This is illustrated by the following example (from the IWSLT 2007 Arabic-English translation task):

Source 1: Asf lA ymknk *lk hnAk klfp HwAly vmAnyn dwlAr lAlsAEp AlwAHdp

Ref: sorry you can't there is a cost the charge is eighty dollars per hour

1-best: i'm sorry you can't there in the cost about eighty dollars for a one o'clock

Source 2: E*rA lA ymknk t$gyl AltlfAz HtY tqlE AlTA}rp

Ref: sorry you cannot turn the tv on until the plane has taken off

1-best: excuse me i you turn tv until the plane departs

The phrase *lA ymknk* (*you may not/you cannot*) is translated differently (and wrongly in the second case) due to different segmentations and phrase translations chosen by the decoder. Though different choices may be sometimes appropriate, the lack of constraints enforcing translation consistency often leads to suboptimal translation performance. It would be desirable to counter this effect by encouraging similar outputs for similar inputs (under a suitably defined notion of similarity, which may include e.g. a context specification for the phrase/sentence).

In machine learning, the idea of forcing the outputs of a statistical learner to vary smoothly with the underlying structure of the inputs has been formalized in the graph-based learning (GBL) framework. In GBL, both labeled (train) and unlabeled (test) data samples are jointly represented as vertices in a graph whose edges encode pairwise similarities between samples. Various learning algorithms can be applied to assign labels to the test samples while ensuring that the classification output varies smoothly

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 119–127,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

along the manifold defined by the graph. GBL has been successfully applied to a range of problems in computer vision, computational biology, and natural language processing. However, in most cases, the learning tasks consisted of unstructured classification, where the input was represented by fixed-length feature vectors and the output was one of a finite set of discrete labels. In machine translation, by contrast, both inputs and outputs consist of word strings of variable length, and the number of possible outputs is not fixed and practically unlimited.

In this paper we propose a new graph-based learning algorithm with structured inputs and outputs to improve consistency in phrase-based statistical machine translation. We define a joint similarity graph over training and test data and use an iterative label propagation procedure to regress a scoring function over the graph. The resulting scores for unlabeled samples (translation hypotheses) are then combined with standard model scores in a log-linear translation model for the purpose of reranking. Our contributions are twofold. First, from a machine translation perspective, we design and evaluate a global consistency model enforcing that similar inputs receive similar translations. Second, from a machine learning perspective, we apply graph-based learning to a task with structured inputs and outputs, which is a novel contribution in itself since previous applications of GBL have focused on predicting categorical labels. We evaluate our approach on two machine translation tasks, the IWSLT 2007 Italian-to-English and Arabic-to-English tasks, and demonstrate significant improvements over the baseline.

2 Graph-Based Learning

GBL algorithms rely on a similarity graph consisting of a set of nodes representing data samples x_i (where i ranges over $1, \ldots, l$ labeled points and $l+1, \ldots, n$ unlabeled points), and a set of weighted edges encoding pairwise similarities between samples. The graph is characterized by a weight matrix W whose elements $W_{ij} \geq 0$ are the similarity values for edges between vertices x_i and x_j, and by its label vector $Y = (y_1, \ldots y_l), y_i \in \{1, \ldots, C\}$ that defines labels for the first l points. If there is no edge linking nodes x_i and x_j, then $W_{ij} = 0$. There is considerable freedom in choosing the weights. The similarity measure used to compute the edge weights determines the graph structure and is the most important factor in successfully applying GBL. In most applications of GBL, data samples are represented by fixed-length feature vectors, and cosine similarity or Euclidean distance-based measures are used for edge weights.

Learning algorithms on similarity graphs include e.g. min-cut (Blum and Chawla, 2001), spectral graph transducer (Joachims, 2003), random walk-based approaches (Szummer and Jaakkola, 2001), and label propagation (Zhu and Ghahramani, 2002). The algorithm proposed herein is based on the latter.

2.1 Label Propagation

Given a graph defined by a weight matrix W and a label set Y, the basic label propagation algorithm proceeds as follows:

1. Initialize the matrix P as $P_{ij} = \frac{W_{ij} - W_{ii}}{\sum_j W_{ij} - W_{ii}}$
2. Initialize a $n \times C$ matrix f with binary vectors encoding the known labels for the first l rows: $f_i = \delta_C(y_i)\ \forall i \in \{1, 2, \ldots, l\}$, where $\delta_C(y_i)$ is the Kronecker vector of length C with 1 in position y_i and 0 elsewhere. The remaining rows of f can be zero.
3. $f' \leftarrow P \times f$
4. Clamp already-labeled data rows: $f'_i = \delta_C(y_i)$ $\forall i \in \{1, 2, \ldots, l\}$
5. If $f' \cong f$, stop.
6. $f \leftarrow f'$
7. Repeat from step 3.

After convergence, f contains the solution in rows $l+1$ to n in the form of unnormalized label probability distributions. Hard labels can be obtained by

$$\hat{y}_i = \underset{j \in \{1,\ldots,C\}}{\arg\max}\ f_{ij} \quad \forall i \in \{l+1, \ldots, n\} \qquad (1)$$

The algorithm minimizes the following cost function (Zhu, 2005):

$$\mathcal{S} = \sum_{k=1}^{C} \sum_{i>l\ \vee\ j>l} W_{ij}(f_{ik} - f_{jk})^2 \qquad (2)$$

$\mathcal{S}$ measures the smoothness of the learned function, i.e., the extent to which the labeling allows large-weight edges to link nodes of different labels. By minimizing $\mathcal{S}$, label propagation finds a labeling

that, to the extent possible, assigns similar soft labels (identical hard labels) to nodes linked by edges with large weights (i.e., highly similar samples). The labeling decision takes into account not only similarities between labeled and unlabeled nodes (as in nearest-neighbor approaches) but also similarities among unlabeled nodes. Label propagation has been used successfully for various classification tasks, e.g. image classification and handwriting recognition (Zhu, 2005). In natural language processing, label propagation has been used for document classification (Zhu, 2005), word sense disambiguation (Niu et al., 2005; Alexandrescu and Kirchhoff, 2007), and sentiment categorization (Goldberg and Zhu, 2006).

3 Graph-Based Learning for Machine Translation

Our goal is to exploit graph-based learning for improving consistency in statistical phrase-based machine translation. Intuitively, a set of similar source sentences should receive similar target-language translations. This means that similarities between training and test sentences should be taken into account, but *also similarities between different test sentences*, which is a source of information currently not exploited by standard SMT systems. To this end we define a graph over the training and test sets with edges between test and training sentences as well as between different test sentences. In cases where a test sentence does not have any connections to training sentences but is connected to other test sentences, helpful information about preferred translations can be propagated via these edges.

As mentioned above, the problem of machine translation does not neatly fit into the standard GBL framework. Given that our samples consist of variable-length word strings instead of feature vectors, the standard cosine or Euclidean-distance based similarity measures cannot be used meaningfully, and the number of possible "labels"—correct translations—is unbounded and practically very large. We thus need to modify both the graph construction and the label propagation algorithms.

First, we handle the problem of unlimited outputs by applying GBL to rescoring only. In most SMT systems, an N-best list (generated by a first decoding pass) approximates the search space of good hypotheses reasonably well, provided N is large enough. For all hypotheses of all sentences in the test set (set we denote with H), the system learns a ranking function $r : H \rightarrow [0, 1]$. Larger values of r indicate better hypotheses. The corresponding loss functional is

$$\mathcal{L}(r) = \sum_{i,j} W_{ij} \left[r(x_i) - r(x_j)\right]^2 \qquad (3)$$

$\mathcal{L}(r)$ measures the smoothness of r over the graph by penalizing highly similar clusters of nodes that have a high variance of r (in other words, similar input sentences that have very different translations). The smaller $\mathcal{L}(r)$, the "smoother" r is over the graph. Thus, instead of directly learning a classification function, we learn a regression function—similar to (Goldberg and Zhu, 2006)—that is then used for ranking the hypotheses.

3.1 Graph Construction

Each graph node represents a sentence *pair* (consisting of source and target strings), and edge weights represent the combined similarity scores computed from comparing both the source sides and target sides of a pair of nodes. Given a training set with l source and target language sentence pairs $(s_1, t_1), \ldots, (s_l, t_l)$ and a test set with $l + 1, ..., n$ source sentences, $s_{l+1}, \ldots, s_n$, the construction of the similarity graph proceeds as follows:

1. For each test sentence $s_i, i = l + 1, \ldots, n$, find a set S_{train_i} of similar training source sentences and a set S_{test_i} of similar test sentences (excluding s_i and sentences identical to it) by applying a string similarity function σ to the source sides only and retaining sentences whose similarity exceeds a threshold θ. Different θ's can be used for training vs. test sentences; we use the same θ for both sets.
2. For each hypothesis h_{s_i} generated for s_i by a baseline system, compute its similarity to the target sides of all sentences in S_{train_i}. The overall similarity is then defined by the combined score

$$\alpha_{ij} = \kappa\left(\sigma(s_i, s^j), \sigma(h_{s_i}, t^j)\right) \qquad (4)$$

where $i = l + 1, \ldots n$, $j = 1, \ldots, |S_{train_i}|$ and $\kappa : \mathbb{R}_+ \times \mathbb{R}_+ \rightarrow \mathbb{R}_+$ is an averaging function.

If $\alpha_{ij} > 0$, establish graph nodes for h_{s_i} and t_j and link them with an edge of weight α_{ij}.

3. For each hypothesis h_{s_i} and each hypothesis generated for each of the sentences $s_k \in \Sigma_{test_i}$, compute similarity on the target side and use the combined similarity score as the edge weight between nodes for h_{s_i} and h_{s_k}.
4. Finally,for each node x_t representing a training sentence, assign $r(x_t) = 1$ and also define its synthetic counterpart: a vertex x'_t with $r(x'_t) = 0$. For each edge incident to x_t of weight W_{th}, define a corresponding edge of weight $1 - W_{t'h}$.

The synthetic nodes and edges need to be added to prevent the label propagation algorithm from converging to the trivial solution that assigns $r = 1$ to all points in the graph. This choice is theoretically motivated—a similarity graph for regression should have not only "sources" (good nodes with high value of r) but also "sinks" (counterparts for the sources). Figure 1 illustrates the connections of a test node.

Similarity Measure The similarity measure used for comparing source and target sides is of prime importance, as it determines the structure of the graph. This has consequences for both computational efficiency (denser graphs require more computation and memory) and the accuracy of the outcome. A low similarity threshold results in a rich graph with a large number of edges but possibly introduces noise. A higher threshold leads to a small graph emphasizing highly similar samples but with too many disconnected components. The similarity measure is also the means by which domain knowledge can be incorporated into the graph construction process. Similarity may be defined at the level of surface word strings, but may also include linguistic information such as morphological features, part-of-speech tags, or syntactic structures. Here, we compare two similarity measures: the familiar BLEU score (Papineni et al., 2002) and a score based on string kernels. In using BLEU we treat each sentence as a complete document. BLEU is not symmetric—when comparing two sentences, different results are obtained depending on which one is considered the reference and which one is the hypothesis. For computing similarities between train and test translations, we use the train translation as the reference. For computing similarity between two test hypotheses, we compute BLEU in both directions and take the average. We note that more appropriate distance measures are certainly possible. Many previous studies, such as (Callison-Burch et al., 2006), have pointed out drawbacks of BLEU, and any other similarity measure could be utilized instead. In particular, similarity measures that model aspects of sentences that are ill handled by standard phrase-based decoders (such as syntactic structure or semantic information) could be useful here.

A more general way of computing similarity between strings is provided by string kernels (Lodhi et al., 2002; Rousu and Shawe-Taylor, 2005), which have been extensively used in bioinformatics and email spam detection. String kernels map strings into a feature space defined by all possible substrings of the string up a fixed length k, and computing the dot product between the resulting feature vectors. Several variants of basic string kernels exist, notably those allowing gaps or mismatches, and efficient implementations have been devised even for large scale applications. Formally, we define a sentence s as a concatenation of symbols from a finite alphabet Σ (the vocabulary of the language) and an embedding function from strings to feature vectors, $\phi : \Sigma^* \rightarrow \mathcal{H}$. A kernel function $\mathcal{K}(s,t)$ computes the distance between the resulting vectors for two sentences s and t. In our case, the embedding function is defined as

$$\phi_u^k(s) := \sum_{\mathbf{i}:u=s(\mathbf{i})} \lambda^{|i|} \quad u \in \Sigma^k \tag{5}$$

where k is the maximum length of substrings, $|i|$ is the length of i, and λ is a penalty parameter for each gap encountered in the substring. $\mathcal{K}$ is defined as

$$\mathcal{K}(s,t) = \sum_u \langle \phi_u(s), \phi_u(t) \rangle w_u \tag{6}$$

where w is a weight dependent on the length of the substring u. Finally, the kernel score is normalized by $\sqrt{\mathcal{K}(s,s) \cdot \mathcal{K}(t,t)}$ to discourage long sentences from being favored. Thus, our similarity measure is a gapped, normalized string kernel, which is a more general measure than BLEU in that is considers noncontiguous substrings. We use a dynamic programming implementation of string kernels (Rousu and Shawe-Taylor, 2005).

For the combination of source-side and target-side similarity scores (the function we denoted as κ) we test two simple schemes, using either the geometric or the arithmetic mean of the individual scores. In the first case, large edge weights only result when both source and target are close to each other; the latter may produce high edge weights when only one of them (typically the source score) is high. More sophisticated combination schemes, using e.g. weighted combination, could be used but were not investigated in this study.

Scalability Poor scalability is often mentioned as a drawback of graph-based learning. Straightforward implementations of GBL algorithms often represent the joint training and test data in working memory and therefore do not scale well to large data sets. However, we have developed several techniques to improve scalability without impeding accuracy. First, we construct separate graphs for each test sentence without losing global connectivity information. The graph for a test sentence is computed as the *transitive closure* of the edge set E over the nodes containing all hypotheses for that test sentence. This smaller graph does not affect the outcome of the learning process for the chosen sentence because in label propagation the learned value $r(x_i)$ can be influenced by that of another node x_j if and only if x_j is reachable from x_i. In the worst theoretical case, the transitive closure could comprehend the entire graph, but in practice the edge set is never that dense and can be easily pruned based on the heuristic that faraway nodes connected through low-weight edges have less influence on the result. We use a simple embodiment of this heuristic in a work-list approach: starting from the nodes of interest (hypotheses for the focal sentence), we expand the closure starting with the direct neighbors, which have the largest influence; then add their neighbors, which have less influence, and so forth. A threshold on the number of added vertices limits undue expansion while capturing either the entire closure or a good approximation of it. Another practical computational advantage of portioning work is that graphs for different hypothesis sets can be trivially created and used in parallel, whereas distributing large matrix-vector multiplication is much more difficult (Choi, 1998). The disadvantage is that overall redundant computations are being made: incomplete estimates of r are computed for the ancillary nodes in the transitive closure and then discarded.

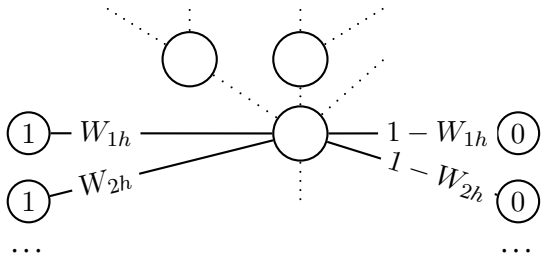


Figure 1: Connections for hypothesis node x_h. Similarity edges with weights W_{th} link the node with train sentences x_t, for which $r(x_t) = 1$. For each of these edges we define a dissimilarity edge of weight $1 - W_{th}$, linking the node with node x'_t for which $r(x'_t) = 0$. The vertex is also connected to other test vertices (the dotted edges).

Second, we obtain a reduction in graph size of orders of magnitude by collapsing all training vertices of the same r that are connected to the same test vertex into one and sum the edge weights. This is equivalent to the full graph for learning purposes.

3.2 Propagation

Label propagation proceeds as follows:

1. Compute the transitive closure over the edges starting from all hypothesis nodes of a given sentence.
2. On the resulting graph, collapse all test-train similarities for each test node by summing edge weights. Obtain accumulated similarities in row and column 1 of the similarity matrix W.
3. Normalize test-to-train weights such that $\sum_j W_{1j} = \sum_j W_{j1} = 1$.
4. Initialize the matrix P as $P_{ij} = \frac{W_{ij}}{1 - W_{i1} + \sum_j W_{ij}}$. (The quantity $1 - W_{1i}$ in the denominator is the weight of the dissimilarity edge.)
5. Initialize a column vector f of height n with $f_1 = 1$ (corresponding to node x_1) and 0 in the remaining positions.
6. $f' \leftarrow P \times f$
7. Clamp f'_1: $f'_1 = 1$
8. If $f' \cong f$, continue with step 11.
9. $f \leftarrow f'$
10. Repeat from step 6.
11. The result r is in the slots of f that correspond to the hypotheses of interest. Normalize per sentence if needed, and rank in decreasing order of r.

Convergence Our algorithm's convergence proof is similar to that for standard label propagation (Zhu, 2005, p. 6). We split P as follows:

$$P = \begin{bmatrix} 0 & P_{LU} \\ P_{UL} & P_{UU} \end{bmatrix} \quad (7)$$

where P_{UL} is a column vector holding global similarities of test hypotheses with train sentences, P_{LU} is a horizontal vector holding the same similarities (though $P_{LU} \neq P_{UL}^T$ due to normalization), and P_{UU} holds the normalized similarities between pairs of test hypotheses. We also separate f:

$$f = \begin{bmatrix} 1 \\ f_U \end{bmatrix} \quad (8)$$

where we distinguish the first entry because it represents the training part of the data. With these notations, the iteration formula becomes:

$$f'_U = P_{UU} f_U + P_{UL} \quad (9)$$

Unrolling the iteration yields:

$$f_U = \lim_{n \to \infty} \left[(P_{UU})^n f_U^0 + \left(\sum_{i=1}^{n} (P_{UU})^{i-1} \right) P_{UL} \right]$$

It can be easily shown that the first term converges to zero because of normalization in step 4 (Zhu, 2005). The sum in the second term converges to $(I - P_{UU})^{-1}$, so the unique fixed point is:

$$f_U = (I - P_{UU})^{-1} P_{UL} \quad (10)$$

Our system uses the iterative form. On the data sets used, convergence took 61.07 steps on average.

At the end of the label propagation algorithm, normalized scores are obtained for each N-best list (sentences without any connections whatsoever are assigned zero scores). These are then used together with the other component models in log-linear combination. Combination weights are optimized on a held-out data set.

4 Data and System

We evaluate our approach on the IWSLT 2007 Italian-to-English (IE) and Arabic-to-English (AE) travel tasks. The first is a challenge task, where the training set consists of read sentences but the development and test data consist of spontaneous dialogues. The second is a standard travel expression translation task consisting entirely of read input. For our experiments we chose the text input (correct transcription) condition only. The data set sizes are shown in Table 1. We split the IE development set into two subsets of 500 and 496 sentences each. The first set (dev-1) is used to train the system parameters of the baseline system and as a training set for GBL. The second is used to tune the GBL parameters. For each language pair, the baseline system was trained with additional out-of-domain text data: the Italian-English Europarl corpus (Koehn, 2005) in the case of the IE system, and 5.5M words of newswire data (LDC Arabic Newswire, Multiple-Translation Corpus and ISI automatically extracted parallel data) in the case of the AE system.

Set	# sent pairs	# words	# refs
IE train	26.5K	160K	1
IE dev-1	500	4308	1
IE dev-2	496	4204	1
IE eval	724	6481	4
AE train	23K	160K	1
AE dev4	489	5392	7
AE dev5	500	5981	7
AE eval	489	2893	6

Table 1: Data set sizes and reference translations count.

Our baseline is a standard phrase-based SMT system based on a log-linear model with the following feature functions: two phrase-based translation scores, two lexical translation scores, word count and phrase count penalty, distortion score, and language model score. We use the Moses decoder (Koehn et al., 2007) with a reordering limit of 4 for both languages, which generates N-best lists of up to 2000 hypotheses per sentence in a first pass. The second pass uses a part-of-speech (POS) based trigram model, trained on POS sequences generated by a MaxEnt tagger (Ratnaparkhi, 1996). The language models are trained on the English side using SRILM (Stolcke, 2002) and modified Kneser-Ney discounting for the first-pass models and Witten-Bell discounting for the POS models. The baseline system yields state-of-the-art performance.

Weighting	dev-2	eval
none (baseline)	22.3/53.3	29.6/45.5
(a)	23.4/51.5	30.7/44.1
(b)	23.5/51.6	30.6/44.3
(c)	23.2/51.8	30.0/44.6

Table 2: GBL results (%BLEU/PER) on IE task for different weightings of labeled-labeled vs. labeled-unlabeled graph edges (BLEU-based similarity measure).

5 Experiments and Results

We started with the IE system and initially investigated the effect of only including edges between labeled and unlabeled samples in the graph. This is equivalent to using a weighted k-nearest neighbor reranker that, for each hypothesis, computes average similarity with its neighborhood of labeled points, and uses the resulting score for reranking.

Starting with the IE task and the BLEU-based similarity metric, we ran optimization experiments that varied the similarity threshold and compared sum vs. product combination of source and target similarity scores, settling for $\theta = 0.7$ and product combination. We experimented with three different ways of weighting the contributions from labeled-unlabeled vs. unlabeled-unlabeled edges: (a) no weighting, (b) labeled-to-unlabeled edges were weighted 4 times stronger than unlabeled-unlabeled ones; and (c) labeled-to-unlabeled edges were weighted 2 times stronger. The weighting schemes do not lead to significantly different results. The best result obtained shows a gain of 1.2 BLEU points on the dev set and 1 point on the eval set, reflecting PER gains of 2% and 1.2%, respectively.

We next tested the string kernel based similarity measure. The parameter values were 0.5 for the gap penalty, a maximum substring length of $k = 4$, and weights of 0, 0.1, 0.2, 0.7. These values were chosen heuristically and were not tuned extensively due to time constraints. Results (Table 3) show significant improvements in PER and BLEU.

In the context of the BTEC challenge task it is interesting to compare this approach to adding the development set directly to the training set. Part of the improvements may be due to utilizing kNN information from a data set that is matched to the test

System	dev-2	eval
Baseline	22.3/53.3	29.6/45.5
GBL	**24.3/51.0**	**32.2/42.7**

Table 3: GBL results (%BLEU/PER) on IE tasks with string-kernel based similarity measure.

set in terms of style. If this data were also used for training the initial phrase table, the improvements might disappear. We first optimized the log-linear model combination weights on the entire dev07 set (dev-1 and dev-2 in Table 1) before retraining the phrase table using the combined train and dev07 data. The new baseline performance (shown in Table 4) is much better than before, due to the improved training data. We then added GBL to this system by keeping the model combination weights trained for the previous system, using the N-best lists generated by the new system, and using the combined train+dev07 set as a train set for selecting similar sentences. We used the GBL parameters that yielded the best performance in the experiments described above. As can be seen from Table 4, GBL again yields an improvement of up to 1.2% absolute in both BLEU and PER.

System	BLEU (%)	PER
Baseline	37.9	38.4
GBL	**39.2**	**37.2**

Table 4: Effect of GBL on IE system trained with matched data (eval set).

For the AE task we used $\theta = 0.5$; however, this threshold was not tuned extensively. Results using BLEU similarity are shown in Table 5. The best result on the eval set yields an improvement of 1.2 BLEU points though only 0.2% reduction in PER. Overall, results seem to vary with parameter settings and nature of the test set (e.g. on dev5, used as a test set, not for optimization, a surprisingly larger improvement in BLEU of 2.7 points is obtained!).

Overall, sentence similarities were observed to be lower for this task. One reason may be that the AE system includes statistical tokenization of the source side, which is itself error-prone in that it can split the same word in different ways depending on the con-

Method	dev4	dev5	eval
Baseline	30.2/43.5	21.9/48.4	37.8/41.8
GBL	30.3/42.5	24.6/48.1	39.0/41.6

Table 5: AE results (%BLEU/PER, $\theta = 0.5$)

text. Since our similarity measure is word-based, this may cause similar sentences to fall below the threshold. The string kernel does not yield any improvement over the BLEU-based similarity measure on this task. One possible improvement would be to use an extended string kernel that can take morphological similarity into account.

Example Below we give an actual example of a translation improvement, showing the source sentence, the 1-best hypotheses of the baseline system and GBL system, respectively, the references, and the translations of similar sentences in the graph neighborhood of the current sentence.

Source: Al+ mE*rp Aymknk {ltqAT Swrp lnA
Baseline: i'm sorry could picture for us
GBL: excuse me could you take a picture of the us
Refs:
excuse me can you take a picture of us
excuse me could you take a photo of us
pardon would you mind taking a photo of us
pardon me could you take our picture
pardon me would you take a picture of us
excuse me could you take a picture of u
Similar sentences:
could you get two tickets for us
please take a picture for me
could you please take a picture of us

6 Related Work

GBL is an instance of semi-supervised learning, specifically transductive learning. A different form of semi-supervised learning (self-training) has been applied to MT by (Ueffing et al., 2007). Ours is the first study to explore a graph-based learning approach. In the machine learning community, work on applying GBL to structured outputs is beginning to emerge. Transductive graph-based regularization has been applied to large-margin learning on structured data (Altun et al., 2005). However, scalability quickly becomes a problem with these approaches; we solve that issue by working on transitive closures as opposed to entire graphs. String kernel representations have been used in MT (Szedmak, 2007) in a kernel regression based framework, which, however, was an entirely supervised framework. Finally, our approach can be likened to a probabilistic implementation of translation memories (Maruyana and Watanabe, 1992; Veale and Way, 1997). Translation memories are (usually commercial) databases of segment translations extracted from a large database of translation examples. They are typically used by human translators to retrieve translation candidates for subsequences of a new input text. Matches can be exact or fuzzy; the latter is similar to the identification of graph neighborhoods in our approach. However, our GBL scheme propagates similarity scores not just from known to unknown sentences but also indirectly, via connections through other unknown sentences. The combination of a translation memory and statistical translation was reported in (Marcu, 2001); however, this is a combination of word-based and phrase-based translation predating the current phrase-based approach to SMT.

7 Conclusion

We have presented a graph-based learning scheme to implement a consistency model for SMT that encourages similar inputs to receive similar outputs. Evaluation on two small-scale translation tasks showed significant improvements of up to 2.6 points in BLEU and 2.8% PER. Future work will include testing different graph construction schemes, in particular better parameter optimization approaches and better string similarity measures. More gains can be expected when using better domain knowledge in constructing the string kernels. This may include e.g. similarity measures that accommodate POS tags or morphological features, or comparisons of the syntax trees of parsed sentence. The latter could be quite easily incorporated into a string kernel or the related tree kernel similarity measure. Additionally, we will investigate the effectiveness of this approach on larger translation tasks.

Acknowledgments This work was funded by NSF grant IIS-032676 and DARPA under Contract No. HR0011-06-C-0023. 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 these agencies.

References

A. Alexandrescu and K. Kirchhoff. 2007. Data-Driven Graph Construction for Semi-Supervised Graph-Based Learning in NLP. In *HLT*.

Y. Altun, D. McAllester, and M. Belkin. 2005. Maximum margin semi-supervised learning for structured variables. In *Proceedings of NIPS 18*.

A. Blum and S. Chawla. 2001. Learning from labeled and unlabeled data using graph mincuts. *Proc. 18th International Conf. on Machine Learning*, pages 19–26.

C. Callison-Burch, M. Osborne, and P. Koehn. 2006. Re-evaluating the role of BLEU in machine translation research. In *Proceedings of EACL*.

Jaeyoung Choi. 1998. A new parallel matrix multiplication algorithm on distributed-memory concurrent computers. *Concurrency: Practice and Experience*, 10(8):655–670.

A. Goldberg and J. Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL Workshop on Graph-based Algorithms for Natural Language Processing*.

T. Joachims. 2003. Transductive learning via spectral graph partitioning. In *Proceedings of ICML*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

P. Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation Summit X*, pages 79–86, Phuket, Thailand.

H. Lodhi, J. Shawe-taylor, and N. Cristianini. 2002. Text classification using string kernels. In *Proceedings of NIPS*.

D. Marcu. 2001. Towards a unified approach to memory- and statistical-based machine translation. In *Proceedings of ACL*.

H. Maruyana and H. Watanabe. 1992. Tree cover search algorithm for example-based translation. In *Proceedings of TMI*, pages 173–184.

Zheng-Yu Niu, Dong-Hong Ji, and Chew Lim Tan. 2005. Word sense disambiguation using label propagation based semi-supervised learning method. In *Proceedings of ACL*, pages 395–402.

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

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proc.of (EMNLP)*.

J. Rousu and J. Shawe-Taylor. 2005. Efficient computation of gap-weighted string kernels on large alphabets. *Journal of Machine Learning Research*, 6:1323–1344.

A. Stolcke. 2002. SRILM—an extensible language modeling toolkit. In *ICSLP*, pages 901–904.

Zhuoran Wang;John Shawe-Taylor;Sandor Szedmak. 2007. Kernel regression based machine translation. In *Proceedings of NAACL/HLT*, pages 185–188. Association for Computational Linguistics.

Martin Szummer and Tommi Jaakkola. 2001. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14. `http://ai.mit.edu/people/szummer/`.

N. Ueffing, G. Haffari, and A. Sarkar. 2007. Transductive learning for statistical machine translation. In *Proceedings of the ACL Workshop on Statistical Machine Translation*.

T. Veale and A. Way. 1997. Gaijin: a template-based bootstrapping approach to example-based machine translation. In *Proceedings of News Methods in Natural Language Processing*.

X. Zhu and Z. Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical report, CMU-CALD-02.

Xiaojin Zhu. 2005. *Semi-Supervised Learning with Graphs*. Ph.D. thesis, Carnegie Mellon University. CMU-LTI-05-192.

Intersecting multilingual data for faster and better statistical translations

Yu Chen[1,2], **Martin Kay**[1,3], **Andreas Eisele**[1,2]
1: Universität des Saarlandes, Saarbrücken, Germany
2: Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Saarbrücken, Germany
3: Stanford University, CA, USA
{yuchen,kay,eisele}@coli.uni-saarland.de

Abstract

In current phrase-based SMT systems, more training data is generally better than less. However, a larger data set eventually introduces a larger model that enlarges the search space for the translation problem, and consequently requires more time and more resources to translate. We argue redundant information in a SMT system may not only delay the computations but also affect the quality of the outputs. This paper proposes an approach to reduce the model size by filtering out the less probable entries based on compatible data in an intermediate language, a novel use of *triangulation*, without sacrificing the translation quality. Comprehensive experiments were conducted on standard data sets. We achieved significant quality improvements (up to 2.3 BLEU points) while translating with reduced models. In addition, we demonstrate a straightforward combination method for more progressive filtering. The reduction of the model size can be up to 94% with the translation quality being preserved.

1 Introduction

Statistical machine translation (SMT) applies machine learning techniques to a bilingual corpus to produce a translation system entirely automatically. Such a scheme has many potential advantages over earlier systems which relied on carefully crafted rules. The most obvious is that it at dramatically reduces cost in human labor and it is able to reach many critical translation rules that are easily overlooked by human being.

SMT systems generally assemble translations by selecting phrases from a large candidate set. Unsupervised learning often introduces a considerable amount of noise into this set as a result of which the selection process becomes more longer and less effective. This paper provides one approach to these problems.

Various filtering techniques, such as (Johnson et al., 2007) and (Chen et al., 2008), have been applied to eliminate a large portion of the translation rules that were judged unlikely to be of value for the current translation. However, these approaches were only able to improve the translation quality slightly. In this paper, we describe a triangulation approach (Kay, 1997) that incorporates multilingual data to improve system efficiency and translation quality at the same time. Most of the previous triangulation approaches (Kumar et al., 2007; Cohn and Lapata, 2007; Filali and Bilmes, 2005; Simard, 1999; Och and Ney, 2001) add information obtained from a third language. In other words, they work with the union of the data from the different languages. In contrast, we work with the intersection of information acquired through a third language. The hope is that the intersection will be more precise and more compact than the union, so that a better result will be obtained more efficiently.

2 Noise in a phrase-based SMT system

The phrases in a translation model are extracted heuristically from a word alignment between the parallel texts in two languages using machine learning techniques. The translation model feature values are stored in the form of a so-called *phrase-table*,

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 128–136,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

while the distortion model is in the *reordering-table*. As we have said models built in this way tend to contain a contains a considerable amount of noise. The phrase-table entries are far less reliable than the lexicons and grammar rules handcrafted for rule-based systems.

The main source of noise in the phrase table is errors from the word alignment process. For example, many function words occur so frequently that they are incorrectly mapped to translations of many function words in the other language to which they are, in fact, unrelated. On the other hand, many words remain unaligned on account of their very low frequency. Another source noise comes from the phrase extraction algorithm itself. The unaligned words are usually attached to aligned sequences In order to achieve longer phrase pairs.

The final selection of entries from the phrase table is based not only on the values assigned to them there, but also to values coming from the language and reordering models, so that entries that receive an initially high value may end up not being preferred.

(1) Sie lieben ihre Kinder nicht.
they love their children not
They don't love their children.

The frequently occurring German negative "*nicht*" in (1). is sometimes difficult for SMT systems to translate into English because it may appear in many positions of a sentence. For instance, it occurs at the end of the sentence in (1). The phrase pairs "*ihre kinder nicht → their children are not*" and "*ihre kinder nicht → their children*" are both likely also to appear in the phrase table and the former has greater estimated probability. However, the language model would preferred the latter in this example because the sentence "*They love their children are not.*" is unlikely to be attested. Accordingly, SMT system may therefore produce the misleading translation in (2).

(2) They love their children.

The system would not produce translations with the opposite meanings if the noisy entries like "*ihre kinder nicht → their children*" were excluded from the translation candidates. Eliminating the noise should help to improve the system's performance, for both efficiency and translation quality.

3 Triangulated filtering

While direct translation and pivot translation through a bridge language presumably introduce noise, in substantially similar amounts, there is no reason to expect the noise in the two systems to correlate strongly. In fact, the noise from such different sources, tends to be quite distinct, whereas the more useful information is often retained. This encourages us to hope that information gathered from various sources will be more reliable overall.

Our plan is to ameliorate the noise problem by constructing a smaller phrase-table by taking the *i*ntersection of a number of sources. We reason that a target phrase is will appear as a candidate translation of a given source phrase, only if it also appears as a candidate translation for some word or phrase in the bridge language mapping to the source phrase. We refer to this triangulation approach as *triangulated phrase-table filtering*.

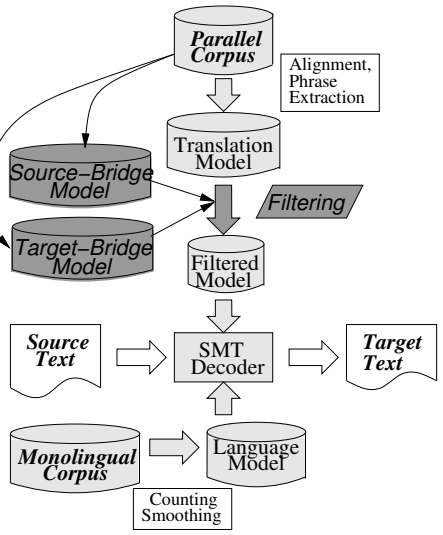


Figure 1: Triangulated filtering in SMT systems

Figure 1 illustrates our triangulation approach. Two bridge models are first constructed: one from the source language to the bridge language, and another from the target language to the bridge language. Then, we use these two models to filter the original source-target model. For each phrase pair in the original table, we try to find a common link in these bridge models to connect both phrases. If such links do not exist, we remove the entry from the table. The probability values in the table remain

unchanged. The reduced table can be used in place of the original one in the SMT system.

There are various forms of links that can be used as our evidence for the filtering process. One obvious form is complete phrases in the bridge language, which means, for each phrase pair in the model to be filtered, we should look for a third phrase in the bridge language that can relate the two phrases in the pair.

This approach to filtering examines each phrase pair presented in the phrase-table one by one. For each phrase pair, we collect the corresponding translations using the models for translation into a third language. If both phrases can be mapped to some phrases in the bridge language, but to different ones, we should remove it from the model. It is also possible that neither of the phrases appear in corresponding bridge models. In this case, we consider the bridge models insufficient for making the filtering decision and prefer to keep the pair in the table.

The way a decoder constructs translation hypotheses is directly related to the weights for different model features in a SMT system, which are usually optimized for a given set of models with minimum error rate training (MERT) (Och, 2003) to achieve better translation performance. In other words, the weights obtained for a model do not necessarily apply to another model. Since the triangulated filtering method removes a part of the model, it is important to readjust the feature weights for the reduced phrase-table.

4 Experimental design

All the text data used in our experiments are from Release v3 of "European Parliament Proceedings Parallel Corpus 1996-2006" (Europarl) corpus (Koehn, 2005). We mainly investigated translations from Spanish to English. There are enough structural differences in these two language to introduce some noise in the phrase table. French, Portuguese, Danish, German and Finnish were used as bridge languages. Portuguese is very similar to Spanish and French somewhat less so. Finnish is unrelated and fairly different typologically with Danish and German occupying the middle ground. In addition, we also present briefly the results on German-English translations with Dutch, Spanish and Danish as bridges.

For the Spanish-English pair, three translation models were constructed over the same parallel corpora. We acquired comparable data sets by drawing several subsets from the same corpus according to various maximal sentence lengths. The subsets

Model	Sentences	Tokens	
		Spanish	English
EP-20	410,487	5,220,142	5,181,452
EP-40	964,687	20,820,067	20,229,833
EP-50	1,100,813	26,731,269	25,867,370
Europarl	1,304,116	37,870,751	36,429,274

Table 1: Europarl subsets for building the Spanish-English SMT system

we used in the experiments are presented by "EP-20", "EP-40" and "EP-50", in which the numbers indicate the maximal sentence length in respective Europarl subsets. Table 1 lists the characteristics of the Spanish-English subsets. Although the maximal sentence length in these sets is far less than that of the whole corpus (880 tokens), EP-50 already includes nearly 85% of Spanish-English sentence pairs from Europarl.

The translations models, both the models to be filtered and the bridge models, were generated from compatible Europarl subsets using the Moses toolkit (Koehn et al., 2007) with the most basic configurations. The feature weights for the Spanish-English translation models were optimized over a development set of 500 sentences using MERT to maximize BLEU (Papineni et al., 2001).

The triangulated filtering algorithm was applied to each combination of a translation model and a third language. The reordering models were also filtered according to the phrase-table. Only those phrase pairs that appeared in the phrase-table remained in the reordering table. We rerun the MERT process solely based on the remaining entries in the filtered tables. Each table is used to translate a set of 2,000 sentences of test data (from the shared task of the third Workshop on Statistical Machine Translation, 2008 [1]). Both the test set and the development data set have been excluded from the training data.

We evaluated the proposed phrase-table filtering

[1] For details, see
http://www.statmt.org/wmt08/shared-task.html

method mainly from two points of view: the *efficiency* of systems with filtered tables and the *quality* of output translations produced by the systems.

5 Results

5.1 System efficiency

Often the question of machine translation is not only how to produce a good translation, but also how to produce it quickly. To evaluate the system efficiency, we measured both storage space and time consumption. For recording the computation time, we run an identical of installation of the decoder with different models and then measure the average execution time for the given translation task.

In Table 2, we give the number of entries in each phrase table (N), and the physical file size of the phrase table (S_{PT}) and the reordering table (S_{RT}) (without any compression or binarization), T_l, the time for the program to load phrase tables and T_t the time to translate the complete test set. We also highlighted the largest and the smallest reduction from each group.

All filtered models showed significant reductions in size. The greatest reduction of model sizes, taking both phrase-table and reordering table into account, is nearly 11 gigabytes for filtering the largest model (EP-50) with a Finnish bridge, which leads to the maximal time saving of 939 seconds, or almost 16 minutes, for translating two thousand sentences.

The reduction rates from two larger models are very close to each other whereas the filtered table scaled down the most significantly on the smallest model (EP-20), which was in fact constructed over a much smaller subset of Europarl corpus, consisting of less than half of the sentences pairs in the other two Europarl subsets. Compared to the larger Europarl subsets, the small data set is expected to produce more errors through training as there is much less relevant data for the machine learning algorithm to correctly extract useful information from. Consequently, there are more noisy entries in this small model, and therefore more entries to be removed. In addition, the filtering is done by exact matching of complete phrases, which presumably happens much less frequently even for correctly paired phrase pairs in the very limited data supplied by the smallest training set. For the same reason, the distinction between different bridge languages was less clear for this small model.

Due to hardware limitation, we are not able to fit the unfiltered phrase tables completely into the memory. Every table was filtered based on the given input so only a small portion of each table was loaded into memory. This may diminish the difference between the original and the filtered table to a certain degree. The relative time consumptionnevertheless agrees with the reduction in size: phrase tables from the smallest model showed the most reduction for both loading the models and processing the translations.

For loading time, we count the time it takes to start and to load the bilingual phrase-tables plus reordering tables and the monolingual language model into the memory. The majority of the loading time for the smallest model, even before filtering, has been used for loading language models and other start-up processes, could not be reduced as much as the reduction on table size.

5.2 Translation quality

Bridge	EP-20	EP-40	EP-50
—	26.62	31.43	31.68
pt	28.40	**32.90**	**33.93**
fr	28.28	32.69	33.47
da	**28.48**	32.47	33.88
de	28.05	32.65	33.13
fi	**28.02**	**31.91**	**33.04**

Table 3: BLEU scores of translations using filtered phrase tables

Efficiency aside, a translation system should be able to produce useful translation. It is important to verify that the filtering approach does not affect the translation quality of the system. Table 3 show the BLEU scores of each translation acquired in the experiments.

Between translation models of different sizes, there are obvious performance gaps. Different bridge languages can cause different effects on performance. However, the translation qualities from a single model are fairly close to each other. We therefore take it that the effect of the triangulation approach is rather *robust* across translation models of different sizes.

	Time		Table Size		
Model+Bridge	T_l (s)	T_t (s)	N	S_{PT} (byte)	S_{RT} (byte)
EP-20+ —	55	3529	7,599,271	953M	717M
EP-20+ pt	**53**	**2826**	**1,712,508** (22.54%)	**198M**	**149M**
EP-20+ fr	48	2702	1,536,056 (20.21%)	172M	131M
EP-20+ da	52	2786	1,659,067 (21.83%)	186M	141M
EP-20+ de	**43**	2732	**1,260,524** (16.59%)	**132M**	**101M**
EP-20+ fi	47	**2670**	1,331,323 (17.52%)	147M	111M
EP-40+ —	65	3673	19,199,807	2.5G	1.9G
EP-40+ pt	**50**	**3091**	8,378,517 (43.64%)	1.1G	1.8G
EP-40+ fr	46	3129	**8,599,708** (44.79%)	**1.1G**	**741M**
EP-40+ da	42	3050	6,716,304 (34.98%)	842M	568M
EP-40+ de	46	3069	6,113,769 (31.84%)	725M	492M
EP-40+ fi	**40**	**2889**	**4,473,483** (23.30%)	**533M**	**353M**
EP-50+ —	140	4130	54,382,715	7.1G	5.4G
EP-50+ pt	78	3410	13,225,654 (24.32%)	1.6G	1.3G
EP-50+ fr	**97**	**3616**	**24,057,849** (44.24%)	**3.0G**	**2.3G**
EP-50+ da	81	3418	12,547,839 (23.07%)	1.5G	1.2G
EP-50+ de	95	3488	15,938,151 (29.31%)	1.9G	1.5G
EP-50+ fi	**71**	**3191**	**7,691,904** (17.75%)	**895M**	**677M**

Table 2: System efficiency: time consumption and phrase-table size

It is obvious that the best systems are usually NOT from the filtered tables that preserved the most entries from the original. All the filtered models showed some improvement in quality with updated model weights. Mostly around 1.5 BLEU points, the increases ranged from 0.36 to 2.25. Table 4 gives a set of translations from the experiments. The unfiltered baseline system inserted the negative by mistake while all the filtered systems are able to avoid this. It indicates that there are indeed noisy entries affecting translation quality in the original table. We were able to achieve better translations by eliminating noisy entries.

The filtering methods indeed tend to remove entries composed of long phrases. Table 5 lists the average length of phrases in several models. Both source phrases and target phrases are taken into account. The best models have shortest phrases on average. Discarding such entries seems to be necessary. This is consistent with the findings in (Koehn, 2003) that phrases longer than three words improve performance little for training corpora of up to 20 million words.

Quality gains appeared to converge in the results across different bridge languages while the original models became larger. Translations generated using large models filtered with different bridge languages are less diverse. Meanwhile, the degradation is less for a larger model. It is reasonable to expect improvements for extremely large models with arbitrary bridge languages. For relatively small models, the selection of bridge languages would be critical for the effect of our approach.

Bridge	EP-20	EP-40	EP-50
—	3.776	4.242	4.335
pt	3.195	3.943	3.740
fr	3.003	3.809	3.947
da	3.005	3.74	3.453
de	2.535	3.501	3.617
fi	2.893	3.521	3.262

Table 5: Average phrase length

5.3 Language clustering

To further understand how the triangulated filtering approach worked and why it could work as it did, we examined a randomly selected phrase table fragment through the experiments. The segment included 10 potential English translations of the same Spanish word "*fabricantes*", the plural form of the word "*fabricante*" (manufacturer).

Table 6 shows the filtering results on a randomly selected segment from the original "EP-40" model, including 10 English translations of the same source

source	Así, se van modificando poco a poco los principios habituales del Estado de derecho por influencia de una concepcin extremista de la lucha con tra las discriminaciones..
ref	thus , the usual principles of the rule of law are being gradually altered under the influence of an extremist approach to combating discrimination.
baseline	we are not changing the usual principles of the rule of law from the influence of an extremist approach in the fight against discrimination.
pt	so , are gradually changing normal principles of the rule of law by influence of an extremist conception of the fight against discrimination.
fr	so , we are gradually changing the usual principles of the rule of law by influence of an extremist conception of the fight against discrimination.
da	so , are gradually changing the usual principles of the rule of law by influence of an extremist conception of the fight against discrimination.
de	thus , we are gradually altering the usual principles of the rule of law by influence of an extremist conception of the fight against discrimination.
fi	so , are gradually changing normal principles of the rule of law by influence of an extremist conception of the fight against discrimination.

Table 4: Examples

fabricantes	pt	fr	da	de	fi	
a manufacturer	✓	✓	✓		✓	4
battalions	✓	✓	✓			3
car manufacturers have						0
car manufacturers	✓	✓	✓	✓	✓	5
makers	✓	✓			✓	3
manufacturer	✓	✓	✓	✓	✓	5
manufacturers	✓	✓	✓	✓	✓	5
producers are		✓	✓	✓		3
producers need						0
producers	✓	✓	✓	✓	✓	5

Table 6: Phrase-table entries before and after filtering a model with different bridges

word "*fabricantes*". ✓ indicates that the corresponding English phrase remained in the table after triangulated filtering with the corresponding bridge language. We also counted the number of tables that included each phrase pair.

Regardless of the bridge language, the triangulated filtering approach had removed those entries that are clearly noise. Meanwhile, entries which are surely correct were always preserved in the filtered tables. The results of using different bridge languages turned out to be consistent on these extreme cases. The 5 filtering processes agreed on six out of ten pairs.

As for the other 4 pairs, the decisions were different using different bridge languages. The remaining entries were always different when the bridge was changed. None of the languages led to the identical eliminations. None of the cases excludes all errors. Apparently, the selection of bridge languages had immediate effects on the filtering results.

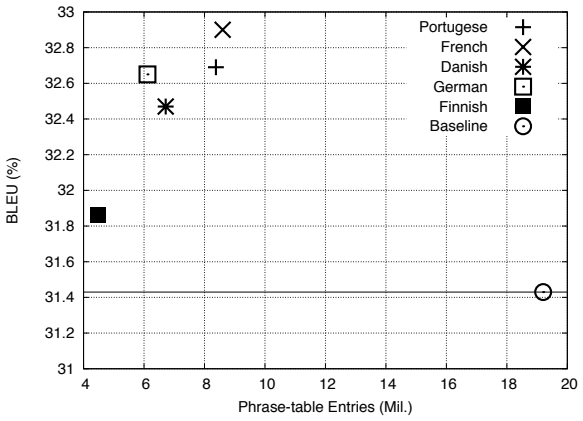


Figure 2: Clustering of bridge languages

We compared two factors of these filtered tables: their sizes and the corresponding BLEU scores. Figure 2 shows interesting signs of language similarity/dissimilarity. There are apparently two groups of languages having extremely close performance, which happen to fall in two language groups: Germanic (German and Danish) and Romance (French and Portuguese). The Romance group was associated with larger filtered tables that produced slightly better translations. The filtered tables created with Germanic bridge languages contained ap-

proximately 2 million entries less than Romance groups. The translation quality difference between these two groups was within 1 point of BLEU.

Observed from this figure, it seems that the translation quality was connected to the similarity between the bridge language and the source language. The closer the bridge is to the source language, the better translations it may produce. For instance, Portuguese led to a filtered table that produced the best translations. On the other hand, the more different the bridge languages compared to the source, the larger portion of the phrase-table the filtering algorithm will remove. The table filtered with German was the smallest in the four cases.

Finnish, a language that is unrelated to others, was associated with distinctive results. The size of the table filtered with Finnish is only 23% of the original, almost half of the table generated with Portuguese. Finnish has extremely rich morphology, hence a great many word-forms, which would make exact matching in bridge models less likely to happen. Many more phrase pairs in the original table were removed for this reason even though some of these entries were beneficial for translations. Even though the improvement on translation quality due to the Finnish bridge was less significant than the others, it is clear that triangulated filtering retained the useful information from the original model.

5.4 Further filtering

The filtering decision with a bridge language on a particular phrase pair is fixed: either to keep the entry or to discard it. It is difficult to adjust the system to work differently. However, as the triangulated filtering procedure does not consider probability distributions in the models, it is possible to further filter the tables according to the probabilities.

The phrase pairs are associated with values computed from the given set of feature weights and sorted, so that we can remove any portions of the remain entries based on the values. Each generated table is used to translate the test set again. Figure 3 shows BLEU scores of the translation outputs produced with tables derived from the "EP-50" model with respect to their sizes. We also included the curve of probability-based filtering alone as the baseline.

The difference between filtered tables at the same

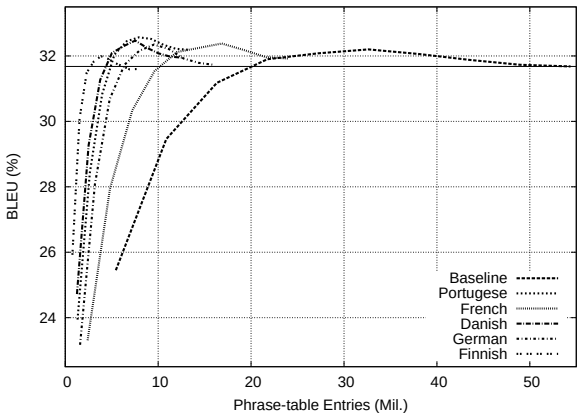


Figure 3: Combining probability-based filtering

size can be over 6 BLEU points, which is a remarkable advantage for the triangulated filtering approach always producing better translations. The curves of the triangulated filtered models are clearly much steeper than that of the naive pruned ones. Data in these filtered models are more compact than the original model before any filtering. The triangulated filtered phrase-tables contain more useful information than a normal phrase-table of the same size. The curves representing the triangulated filtering performance are always on the left of the original curves.

We are able to use less than 6% of the original phrase table (40% of the table filtered with Finnish) to obtain translations with the same quality as the original. The extreme case, using only 1.4% of the original table, leads to a reasonable BLEU score, indicating that most of the output sentences should still be understandable. In this case, the overall size of the phrase table and the reordering table was less than 100 megabytes, potentially feasible for mobile devices, whereas the original models took nearly 12.5 gigabytes of disk space.

5.5 Different source language

Bridge	EP-40		EP-50	
—	5.1G	26.92	6.5G	27.23
Dutch	562M	27.11	1.3G	28.14
Spanish	3.0G	27.28	3.6G	28.09
Danish	505M	28.04	780M	28.21

Table 7: Filtered German-English systems (Size and BLEU)

In addition to Spanish-English translation, we also conducted experiments on German-English translation. The results, shown in Table 7, appear consistent with the results of Spanish-English translation. Translations in most cases have performance close to the original unfiltered models, whereas the reduction in phrase-table size ranged from 40% to 85%. Meanwhile, translation speed has been increased up to 17%.

Due to German's rich morphology, the unfiltered German-English models contain many more entries than the Spanish-English ones constructed from similar data sets. Unlike the Spanish-English models, the difference between "EP-40" and "EP-50" was not significant. Neither was the difference between the impacts of the filtering in terms of translation quality. In addition, German and English are so dissimilar that none of the three bridge languages we chose turned out to be significantly superior.

6 Conclusions

We highlighted one problem of the state-of-the-art SMT systems that was generally neglected: the noise in the translation models. Accordingly, we proposed triangulated filtering methods to deal with this problem. We used data in a third language as evidence to locate the less probable items in the translation models so as to obtain the **intersection** of information extracted from multilingual data. Only the occurrences of complete phrases were taken into account. The probability distributions of the phrases have not been considered so far.

Although the approach was fairly naive, our experiments showed it to be effective. The approaches were applied to SMT systems built with the Moses toolkit. The translation quality was improved at least 1 BLEU for all 15 cases (filtering 3 different models with 5 bridge languages). The improvement can be as much as 2.25 BLEU. It is also clear that the best translations were not linked to the largest translation models. We also sketched a simple extension to the triangulated filtering approach to further reduce the model size, which allows us to generate reasonable results with only 1.4% of the entries from the original table.

The results varied for different bridge languages as well as different models. For translation from Spanish to English, Finnish, the most distinctive bridge language, appeared to be a more effective intermediate language which could remove more phrase pair entries while still improving the translation quality. Portuguese, the most close to the source language, always leads to a filtered model that produces the best translations. The selection of bridge languages has more obvious impact on the performance of our approach when the size of the model to filter was larger.

The work gave one instance of the general approach described in Section 3. There are several potential directions for continuing this work. The most straightforward one is to use our approaches with more different languages, such as Chinese and Arabic, and incompatible corpora, for example, different segments of Europarl. The main focus of such experiments should be verifying the conclusions we had in this paper.

Acknowledgments

This work was supported by European Community through the EuroMatrix project funded under the Sixth Framework Programme and the EuroMatrix Plus project funded under the Seventh Framework Programme for Research and Technological Development.

References

Yu Chen, Andreas Eisele, and Martin Kay. 2008. Improving Statistical Machine Translation Efficiency by Triangulation. In *the 6th International Conference on Language Resources and Evaluation (LREC '08)*, May.

Trevor Cohn and Mirella Lapata. 2007. Machine Translation by Triangulation: Making Effective Use of Multi-Parallel Corpora. In *the 45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech, June.

Karim Filali and Jeff Bilmes. 2005. Leveraging Multiple Languages to Improve Statistical MT Word Alignments. In *IEEE Automatic Speech Recognition and Understanding (ASRU)*, Cancun, Mexico, November.

J. Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. 2007. Improving Translation Quality by Discarding Most of the Phrasetable. In *the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural*

Language Learning (EMNLP-CoNLL), Prague, Czech Republic, June.

Martin Kay. 1997. The proper place of men and machines in language translation. *Machine Translation*, 12(1-2):3–23.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic, June.

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

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *MT Summit 2005*.

Shankar Kumar, Franz Josef Och, and Wolfgang Macherey. 2007. Improving word alignment with bridge languages. In *the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 42–50, Prague, Czech.

Franz Josef Och and Hermann Ney. 2001. Statistical multi-source translation. In *MT Summit VIII*, Santiago de Compostela, Spain.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 160–167, Morristown, NJ, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Michel Simard. 1999. Text-translation alignment: Three languages are better than two. In *EMNLP/VLC-99*, College Park, MD, June.

Without a 'doubt'? Unsupervised discovery of downward-entailing operators

Cristian Danescu-Niculescu-Mizil, Lillian Lee, and Richard Ducott
Department of Computer Science
Cornell University
Ithaca, NY 14853-7501
cristian@cs.cornell.edu, llee@cs.cornell.edu, rad47@cornell.edu

Abstract

An important part of textual inference is making deductions involving *monotonicity*, that is, determining whether a given assertion entails restrictions or relaxations of that assertion. For instance, the statement '*We know the epidemic spread quickly*' does not entail '*We know the epidemic spread quickly via fleas*', but '*We doubt the epidemic spread quickly*' entails '*We doubt the epidemic spread quickly via fleas*'. Here, we present the first algorithm for the challenging lexical-semantics problem of learning linguistic constructions that, like '*doubt*', are *downward entailing (DE)*. Our algorithm is unsupervised, resource-lean, and effective, accurately recovering many DE operators that are missing from the hand-constructed lists that textual-inference systems currently use.

1 Introduction

Making inferences based on natural-language statements is a crucial part of true natural-language understanding, and thus has many important applications. As the field of NLP has matured, there has been a resurgence of interest in creating systems capable of making such inferences, as evidenced by the activity surrounding the ongoing sequence of "Recognizing Textual Entailment" (RTE) competitions (Dagan, Glickman, and Magnini, 2006; Bar-Haim, Dagan, Dolan, Ferro, Giampiccolo, Magnini, and Szpektor, 2006; Giampiccolo, Magnini, Dagan, and Dolan, 2007) and the AQUAINT knowledge-based evaluation project (Crouch, Saurí, and Fowler, 2005).

The following two examples help illustrate the particular type of inference that is the focus of this paper.

1. '*We know the epidemic spread quickly*'
2. '*We doubt the epidemic spread quickly*'

A relaxation of '*spread quickly*' is '*spread*'; a restriction of it is '*spread quickly via fleas*'. From statement 1, we can infer the relaxed version, '*We know the epidemic spread*', whereas the restricted version, '*We know the epidemic spread quickly via fleas*', does not follow. But the reverse holds for statement 2: it entails the restricted version '*We doubt the epidemic spread quickly via fleas*', but not the relaxed version. The reason is that '*doubt*' is a *downward-entailing operator*;[1] in other words, it allows one to, in a sense, "reason from sets to subsets" (van der Wouden, 1997, pg. 90).

Downward-entailing operators are not restricted to assertions about belief or to verbs. For example, the preposition '*without*' is also downward entailing: from '*The applicants came without payment or waivers*' we can infer that all the applicants came without payment. (Contrast this with '*with*', which, like '*know*', is *upward entailing*.) In fact, there are many downward-entailing operators, encompassing many syntactic types; these include explicit negations like '*no*' and '*never*', but also many other terms, such as '*refuse (to)*', '*preventing*', '*nothing*', '*rarely*', and '*too [adjective] to*'.

[1] Synonyms for "downward entailing" include *downward-monotonic* and *monotone decreasing*. Related concepts include *anti-additivity*, *veridicality*, and *one-way implicatives*.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 137–145,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

As the prevalence of these operators indicates and as van der Wouden (1997, pg. 92) states, downward entailment "plays an extremely important role in natural language" (van Benthem, 1986; Hoeksema, 1986; Sánchez Valencia, 1991; Dowty, 1994; MacCartney and Manning, 2007). Yet to date, only a few systems attempt to handle the phenomenon in a general way, i.e., to consider more than simple direct negation (Nairn, Condoravdi, and Karttunen, 2006; MacCartney and Manning, 2008; Christodoulopoulos, 2008; Bar-Haim, Berant, Dagan, Greental, Mirkin, Shnarch, and Szpektor, 2008). These systems rely on lists of items annotated with respect to their behavior in "polar" (positive or negative) environments. The lists contain a relatively small number of downward-entailing operators, at least in part because they were constructed mainly by manual inspection of verb lists (although a few non-verbs are sometimes also included). *We therefore propose to automatically learn downward-entailing operators*[2] *— henceforth DE operators for short — from data; deriving more comprehensive lists of DE operators in this manner promises to substantially enhance the ability of textual-inference systems to handle monotonicity-related phenomena.*

Summary of our approach There are a number of significant challenges to applying a learning-based approach. First, to our knowledge there do not exist DE-operator-annotated corpora, and moreover, relevant types of semantic information are "not available in or deducible from any public lexical database" (Nairn et al., 2006). Also, it seems there is no simple test one can apply to all possible candidates; van der Wouden (1997, pg. 110) remarks, "As a rule of thumb, assume that everything that feels negative, and everything that [satisfies a condition described below], is monotone decreasing. This rule of thumb will be shown to be wrong as it stands; but it sort of works, like any rule of thumb."

Our first insight into how to overcome these challenges is to leverage a finding from the linguistics literature, *Ladusaw's (1980) hypothesis*, which can be treated as a cue regarding the distribution of DE operators: it asserts that a certain class of lexical constructions known as *negative polarity items (NPIs)* can only appear in the scope of DE operators. Note that this hypothesis suggests that one can develop an *unsupervised* algorithm based simply on checking for co-occurrence with known NPIs.

But there are significant problems with applying this idea in practice, including: (a) there is no agreed-upon list of NPIs; (b) terms can be ambiguous with respect to NPI-hood; and (c) many non-DE operators tend to co-occur with NPIs as well. To cope with these issues, we develop a novel unsupervised *distillation* algorithm that helps filter out the noise introduced by these problems. This algorithm is very effective: it is accurate and derives many DE operators that do not appear on pre-existing lists.

Contributions Our project draws a connection between the creation of textual entailment systems and linguistic inquiry regarding DE operators and NPIs, and thus relates to both language-engineering and linguistic concerns.

To our knowledge, this work represents the first attempt to aid in the process of *discovering* DE operators, a task whose importance we have highlighted above. At the very least, our method can be used to provide high-quality raw materials to help human annotators create more extensive DE operator lists. In fact, while previous manual-classification efforts have mainly focused on verbs, we retrieve DE operators across multiple parts of speech. Also, although we discover many items (including verbs) that are not on pre-existing manually-constructed lists, the items we find occur frequently — they are not somehow peculiar or rare.

Our algorithm is surprisingly accurate given that it is quite resource- and knowledge-lean. Specifically, it relies only on Ladusaw's hypothesis as initial inspiration, a relatively short and arguably noisy list of NPIs, and a large unannotated corpus. It does *not* use other linguistic information — for example, we do not use parse information, even though c-command relations have been asserted to play a

[2] We include superlatives ('*tallest*'), comparatives ('*taller*'), and conditionals ('*if*') in this category because they have non-default (i.e., non-upward entailing) properties — for instance, '*he is the tallest father*' does not entail '*he is the tallest man*'. Thus, they also require special treatment when considering entailment relations. In fact, there have been some attempts to unify these various types of non-upward entailing operators (von Fintel, 1999). We use the term *downward entailing (narrowly-defined) (DE(ND))* when we wish to specifically exclude superlatives, comparatives, and conditionals.

key role in the licensing of NPIs (van der Wouden, 1997).

2 Method

We mentioned in the introduction some significant challenges to developing a machine-learning approach to discovering DE operators. The key insight we apply to surmount these challenges is that in the linguistics literature, it has been hypothesized that there is a strong connection between DE operators and *negative polarity items (NPIs)*, which are terms that tend to occur in "negative environments". An example NPI is '*anymore*': one can say '*We don't have those anymore*' but not '$*$*We have those anymore*'.

Specifically, we propose to take advantage of the seminal hypothesis of Ladusaw (1980, influenced by Fauconnier (1975), inter alia):

> (Ladusaw) NPIs only appear within the scope of downward-entailing operators.

This hypothesis has been actively discussed, updated, and contested by multiple parties (Linebarger, 1987; von Fintel, 1999; Giannakidou, 2002, inter alia). It is not our intent to comment (directly) on its overall validity. Rather, we simply view it as a very useful starting point for developing computational tools to find DE operators— indeed, even detractors of the theory have called it "impressively algorithmic" (Linebarger, 1987, pg. 361).

First, a word about scope. For Ladusaw's hypothesis, scope should arguably be defined in terms of c-command, immediate scope, and so on (von Fintel, 1999, pg. 100). But for simplicity and to make our approach as resource-lean as possible, we simply assume that potential DE operators occur to the left of NPIs,[3] except that we ignore text to the left of any preceding commas or semi-colons as a way to enforce a degree of locality. For example, in both '*By the way, we don't have plants anymore*$_{\text{NPI}}$ *because they died*' and '*we don't have plants anymore*$_{\text{NPI}}$', we look for DE operators within the sequence of words '*we don't have plants*'. We refer to such sequences in which we seek DE operators as *NPI contexts*.

[3]There are a few exceptions, such as with the NPI "for the life of me" (Hoeksema, 1993).

Now, Ladusaw's hypothesis suggests that we can find DE operators by looking for words that tend to occur more often in NPI contexts than they occur overall. We formulate this as follows:

> *Assumption:* For any DE operator d, $F_{\text{byNPI}}(d) > F(d)$.

Here, $F_{\text{byNPI}}(d)$ is the number of occurrences of d in NPI contexts[4] divided by the number of words in NPI contexts, and $F(x)$ refers to the number of occurrences of x relative to the number of words in the corpus.

An additional consideration is that we would like to focus on the discovery of *novel* or non-obvious DE operators. Therefore, for a given candidate DE operator c, we compute $\widehat{F}_{\text{byNPI}}(c)$: the value of $F_{\text{byNPI}}(c)$ that results *if we discard all NPI contexts containing a DE operator on a list of 10 well-known instances*, namely, '*not*', '*n't*', '*no*', '*none*', '*neither*', '*nor*', '*few*', '*each*', '*every*', and '*without*'. (This list is based on the list of DE operators used by the RTE system presented in MacCartney and Manning (2008).) This yields the following scoring function:

$$S(c) := \frac{\widehat{F}_{\text{byNPI}}(c)}{F(c)}. \quad (1)$$

Distillation There are certain terms that are not DE operators, but nonetheless co-occur with NPIs as a side-effect of co-occurring with true DE operators themselves. For instance, the proper noun '*Milken*' (referring to Michael Milken, the so-called "junk-bond king") occurs relatively frequently with the DE operator '*denies*', and '*vigorously*' occurs frequently with DE operators like '*deny*' and '*oppose*'. We refer to terms like '*milken*' and '*vigorously*' as "piggybackers", and address the piggybackers problem by leveraging the following intuition: in general, we do not expect to have two DE operators in the same NPI context.[5] One way to implement this would be to re-score the candidates in a winner-takes-all fashion: for each NPI context, reward only the candidate

[4]Even if d occurs multiple times in a single NPI context we only count it once; this way we "dampen the signal" of function words that can potentially occur multiple times in a single sentence.

[5]One reason is that if two DE operators are composed, they ordinarily create a positive context, which would not license NPIs (although this is not always the case (Dowty, 1994)).

with the highest score S. However, such a method is too aggressive because it would force us to pick a single candidate even when there are several with relatively close scores — and we know our score S is imperfect. Instead, we propose the following "soft" mechanism. Each sentence distributes a "budget" of total score 1 among the candidates it contains according to the relative scores of those candidates; this works out to yield the following new *distilled* scoring function

$$S_d(c) = \frac{\sum_{\text{NPIcontexts } p} \frac{S(c)}{n(p)}}{N(c)}, \qquad (2)$$

where $n(p) = \sum_{c \in p} S(c)$ is an NPI-context normalizing factor and $N(c)$ is the number of NPI contexts containing the candidate c. This way, plausible candidates that have high S scores relative to the other candidates in the sentence receive enhanced S_d scores. To put it another way: apparently plausible candidates that often appear in sentences with multiple good candidates (i.e., piggybackers) receive a low distilled score, despite a high initial score.

Our general claim is that the higher the distilled score of a candidate, the better its chances of being a DE operator.

Choice of NPIs Our proposed method requires access to a set of NPIs. However, there does not appear to be universal agreement on such a set. Lichte and Soehn (2007) mention some doubts regarding approximately 200 (!) of the items on a roughly 350-item list of German NPIs (Kürschner, 1983). For English, the "moderately complete"[6] Lawler (2005) list contains two to three dozen items; however, there is also a list of English NPIs that is several times longer (von Bergen and von Bergen, 1993, written in German), and Hoeksema (1997) asserts that English should have hundreds of NPIs, similarly to French and Dutch.

We choose to focus on the items on these lists that seem most likely to be effective cues for our task. Specifically, we select a subset of the Lawler NPIs, focusing mostly on those that do not have a relatively frequent non-NPI sense. An example discard is '*much*', whose NPI-hood depends on what it modifies and perhaps on whether there are degree adverbs pre-modifying it (Hoeksema, 1997). There are some ambiguous NPIs that we do retain due to their frequency. For example, '*any*' occurs both in a non-NPI "free choice" variant, as in '*any idiot can do that*', and in an NPI version. Although it is ambiguous with respect to NPI-hood, '*any*' is also a very valuable cue due to its frequency.[7] Here is our NPI list:

any	in weeks/ages/years	budge	yet
at all	drink a drop	red cent	ever
give a damn	last/be/take long	but what	bother to
do a thing	arrive/leave until	give a shit	lift a finger
bat an eye	would care/mind	eat a bite	to speak of

3 Experiments

Our main set of evaluations focuses on the precision of our method at discovering new DE operators. We then briefly discuss evaluation of other dimensions.

3.1 Setup

We applied our method to the entirety of the BLLIP (Brown Laboratory for Linguistic Information Processing) 1987–89 WSJ Corpus Release 1, available from the LDC (LDC2000T43). The 1,796,379 sentences in the corpus comprise 53,064 NPI contexts; after discarding the ones containing the 10 well-known DE operators, 30,889 NPI contexts were left. To avoid sparse data problems, we did not consider candidates with very low frequency in the corpus ($\leqslant$150 occurrences) or in the NPI contexts ($\leqslant$10 occurrences).

Methodology for eliciting judgments The obvious way to evaluate the precision of our algorithm is to have human annotators judge each output item as to whether it is a DE operator or not. However, there are some methodological issues that arise.

First, if the judges know that every term they are rating comes from our system and that we are hoping that the algorithm extracts DE operators, they may be biased towards calling every item "DE" regardless of whether it actually is. We deal with this problem by introducing *distractors* — items that are not produced by our algorithm, but are similar enough to not be easily identifiable as "fakes". Specifically,

[6] www-personal.umich.edu/~jlawler/aue/npi.html

[7] It is by far the most frequent NPI, appearing in 36,554 of the sentences in the BLLIP corpus (see Section 3).

for each possible part of speech of each of our system's outputs c that exists in WordNet, we choose a distractor that is either in a "sibling" synset (a hyponym of c's hypernym) or an antonym. Thus, the distractors are highly related to the candidates. Note that they may in fact also be DE operators.

The judges were made aware of the presence of a substantial number of distractors (about 70 for the set of top 150 outputs). This design choice did seem to help ensure that the judges carefully evaluated each item.

The second issue is that, as mentioned in the introduction, there does not seem to be a uniform test that judges can apply to all items to ascertain their DE-ness; but we do not want the judges to improvise excessively, since that can introduce undesirable randomness into their decisions. We therefore encouraged the judges to try to construct sentences wherein the arguments for candidate DE operators were drawn from a set of phrases and restricted replacements we specified (example: '*singing*' vs '*singing loudly*'). However, improvisation was still required in a number of cases; for example, the candidate '*act*', as either a noun or a verb, cannot take '*singing*' as an argument.

The labels that the judges could apply were "DE(ND)" (downward entailing (narrowly-defined)), "superlative", "comparative", "conditional", "hard to tell", and "not-DE" (= none of the above). We chose this fine-grained sub-division because the second through fourth categories are all known to co-occur with NPIs. There is some debate in the linguistics literature as to whether they can be considered to be downward entailing, narrowly construed, or not (von Fintel, 1999, inter alia), but nonetheless, such operators call for special reasoning quite distinct from that required when dealing with upward entailing operators — hence, we consider it a success when our algorithm identifies them.

Since monotonicity phenomena can be rather subtle, the judges engaged in a collaborative process. Judge A (the second author) annotated all items, but worked in batches of around 10 items. At the end of each batch, Judge B (the first author) reviewed Judge A's decisions, and the two consulted to resolve disagreements as far as possible.

One final remark regarding the annotation: some decisions still seem uncertain, since various factors such as context, Gricean maxims, what should be presupposed[8] and so on come into play. However, we take comfort in a comment by Eugene Charniak (personal communication) to the effect that if a word causes a native speaker to pause, that word is interesting enough to be included. And indeed, it seems reasonable that if a native speaker thinks there might be a sense in which a word can be considered downward entailing, then our system should flag it as a word that an RTE system should at least perhaps pass to a different subsystem for further analysis.

3.2 Precision Results

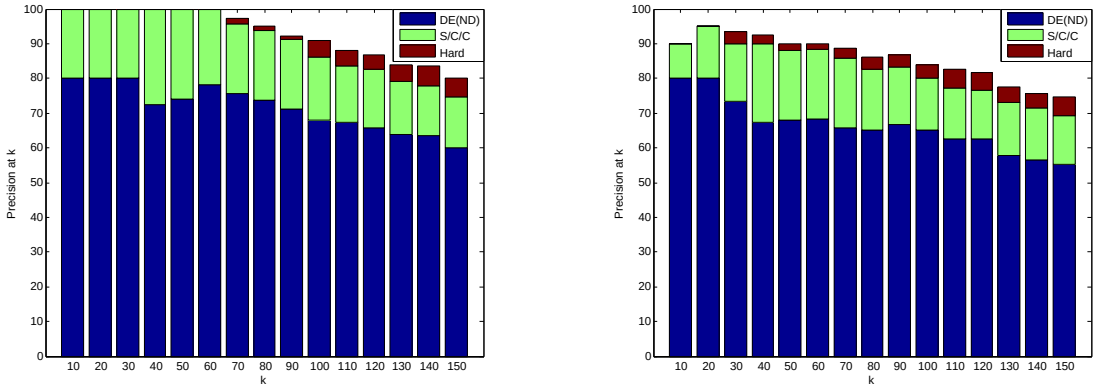
We now examine the 150 items that were most highly ranked by our system, which were subsequently annotated as just described. (For full system output that includes the unannotated items, see `http://www.cs.cornell.edu/~cristian`. We would welcome external annotation help.) As shown in Figure 1a, which depicts precision at k for various values of k, our system performs very well. In fact, *100%* of the first 60 outputs are DE, broadly construed. It is also interesting to note the increasing presence of instances that the judges found hard to categorize as we move further down the ranking.

Of our 73 distractors, 46% were judged to be members of one of our goal categories. The fact that this percentage is substantially lower than our algorithm's precision at both 73 and 150 (the largest k we considered) confirms that our judges were not making random decisions. (We expect the percentage of DE operators among the distractors to be much higher than 0 because they were chosen to be similar to our system's outputs, and so can be expected to also be DE operators some fraction of the time.)

Table 1 shows the lemmas of just the DE(ND) operators that our algorithm placed in its top 150 outputs.[9] Most of these lemmas are new discoveries, in the sense of not appearing in Ladusaw's (1980) (implicit) enumeration of DE operators. Moreover, the

[8] For example, '*X doubts the epidemic spread quickly*' might be said to entail '*X would doubt the epidemic spreads via fleas, presupposing that X thinks about the flea issue*'.

[9] By listing lemmas, we omit variants of the same word, such as '*doubting*' and '*doubted*', to enhance readability. We omit superlatives, comparatives, and conditionals for brevity.



(a) (b)

Figure 1: (a) Precision at k for k divisible by 10 up to $k = 150$. The bar divisions are, from the x-axis up, DE(ND) (blue, the largest); Superlatives/Conditionals/Comparatives (green, 2nd largest); and Hard (red, sometimes non-existent). For example, all of the first 10 outputs were judged to be either downward entailing (narrowly-defined) (8 of 10, or 80%) or in one of the related categories (20%). (b) Precision at k when the distillation step is omitted.

not-DE			Hard
almost	firmly	one-day	approve
ambitious	fined	signal	cautioned
considers	liable	remove	dismissed
detect	notify	vowed	fend

Table 3: Examples of words judged to be either not in one of our monotonicity categories of interest (not-DE) or hard to evaluate (Hard).

lists of DE(ND) operators that are used by textual-entailment systems are significantly smaller than that depicted in Table 1; for example, MacCartney and Manning (2008) use only about a dozen (personal communication).

Table 3 shows examples of the words in our system's top 150 outputs that are either clear mistakes or hard to evaluate. Some of these are due to idiosyncrasies of newswire text. For instance, we often see phrases like '*biggest one-day drop in ...*', where '*one-day*' piggybacks on superlatives, and '*vowed*' piggybacks on the DE operator '*veto*', as in the phrase '*vowed to veto*'.

Effect of distillation In order to evaluate the importance of the distillation process, we study how the results change when distillation is omitted (thus using as score function S from Equation 1 rather than S_d). When comparing the results (summarized in Figure 1b) with those of the complete system (Figure 1a) we observe that the distillation indeed has the desired effect: the number of highly ranked words that are annotated as not-DE decreases after distillation. This results in an increase of the precision at k ranging from 5% to 10% (depending on k), as can be observed by comparing the height of the composite bars in the two figures.[10]

Importantly, this improvement does indeed seem to stem at least in part from the distillation process handling the piggybacking problem. To give just a few examples: '*vigorously*' is pushed down from rank 48 (undistilled scoring) to rank 126 (distilled scoring), '*one-day*' from 25^{th} to 65^{th}, '*vowed*' from 45^{th} to 75^{th}, and '*Milken*' from 121^{st} to 350^{th}.

3.3 Other Results

It is natural to ask whether the (expected) decrease in precision at k is due to the algorithm assigning relatively low scores to DE operators, so that they do not appear in the top 150, or due to there being no more more true DE operators to rank. We cannot directly evaluate our method's recall because no comprehensive list of DE operators exists. However, to get a rough impression, we can check how our system ranks the items in the largest list we are aware of, namely, the Ladusaw (implicit) list mentioned above. Of the 31 DE operator lemmas on this list (not including the 10 well-known DE operators), only 7 of those frequent enough to be considered by our algorithm are not in its top 150 outputs, and only

[10] The words annotated "hard" do not affect this increase in precision.

absence of	to defer	hardly (**L**)	premature to	to rule out	to veto
absent from	to deny (**L**)	to lack	to prevent	skeptical •	wary of
anxious about •	to deter	innocent of •	to prohibit	to suspend	warned that (**L**)
to avoid (**L**)	to discourage	to minimize •	rarely (**L**)	to thwart	whenever
to bar	to dismiss	never (**L**)	to refrain from	unable to	withstand
barely	to doubt (**L**)	nobody	to refuse (**L**)	unaware of	
to block	to eliminate	nothing	regardless •	unclear on	
cannot (**L**)	essential for •	to oppose	to reject	unlike	
compensate for •	to exclude	to postpone •	reluctant to (**L**)	unlikely (**L**)	
to decline	to fail (**L**)	to preclude	to resist	unwilling to	

Table 1: The 55 lemmas for the 90 downward entailing (narrowly-defined) operators among our algorithm's top 150 outputs. (**L**) marks instances from Ladusaw's list. • marks some of the more interesting cases. We have added function words (e.g., '*to*', '*for*') to indicate parts of speech or subcategorization; our algorithm does not discover multi-word phrases.

Original		**Restriction**
Dan is unlikely to sing.	$\Rightarrow$ $\not\Leftarrow$	Dan is unlikely to sing loudly.
Olivia compensates for eating by exercising.	$\Rightarrow$ $\not\Leftarrow$	Olivia compensates for eating late by exercising.
Ursula refused to sing or dance.	$\Rightarrow$ $\not\Leftarrow$	Ursula refused to sing.
Bob would postpone singing.	$\Rightarrow$ $\not\Leftarrow$	Bob would postpone singing loudly.
Talent is essential for singing.	$\Rightarrow$ $\not\Leftarrow$	Talent is essential for singing a ballad.
She will finish regardless of threats.	$\Rightarrow$ $\not\Leftarrow$	She will finish regardless of threats to my career.

Table 2: Example demonstrations that the underlined expressions (selected from Table 1) are DE operators: the sentences on the left entail those on the right. We also have provided $\not\Leftarrow$ indicators because the reader might find it helpful to reason in the opposite direction and see that these expressions are not upward entailing.

5 are not in the top 300. Remember that we only annotated the top 150 outputs; so, there may be many other DE operators between positions 150 and 300.

Another way of evaluating our method would be to assess the effect of our newly discovered DE operators on downstream RTE system performance. There are two factors to take into account. First, the DE operators we discovered are quite prevalent in naturally occurring text[11] : the 90 DE(ND) operators appearing in our algorithm's top 150 outputs occur in 111,456 sentences in the BLLIP corpus (i.e., in 6% of its sentences). Second, as previously mentioned, systems do already account for monotonicity to some extent — but they are limited by the fact that their DE operator lexicons are restricted mostly to well-known instances; to take a concrete example with a publicly available RTE system: Nutcracker (Bos and Markert, 2006) correctly infers that '*We did not know the disease spread*' entails '*We did not know the disease spread quickly*' but it fails to infer that '*We doubt the disease spread*' entails '*We doubt the disease spread quickly*'. So, systems can use monotonicity information but currently do not have enough of it; our method can provide them with this information, enabling them to handle a greater fraction of the large number of naturally occurring instances of this phenomenon than ever before.

[11]However, RTE competitions do not happen to currently stress inferences involving monotonicity. The reasons why are beyond the scope of this paper.

4 Related work not already discussed

Magnini (2008), in describing modular approaches to textual entailment, hints that NPIs may be used within a negation-detection sub-component.

There is a substantial body of work in the linguistics literature regarding the definition and nature of polarity items (Polarity Items Bibliography). However, very little of this work is computational. There has been passing speculation that one might want to learn polarity-inverting verbs (Christodoulopoulos, 2008, pg. 47). There have also been a few projects on the discovery of NPIs, which is the converse of the problem we consider. Hoeksema (1997) discusses some of the difficulties with corpus-based determination of NPIs, including "rampant" poly-

semy and the problem of "how to determine independently which predicates should count as negative" — a problem which our work addresses. Lichte and Soehn (Lichte, 2005; Lichte and Soehn, 2007) consider finding German NPIs using a method conceptually similar in some respects to our own, although again, their objective is the reverse of ours. Their discovery statistic for single-word NPIs is the ratio of within-licenser-clause occurrences to total occurrences, where, to enhance precision, the list of licensers was filtered down to a set of fairly unambiguous, easily-identified items. They do not consider distillation, which we found to be an important component of our DE-operator-detection algorithm. Their evaluation scheme, unlike ours, did not employ a bias-compensation mechanism. They did employ a collocation-detection technique to extend their list to multi-word NPIs, but our independent experiments with a similar technique (not reported here) did not yield good results.

5 Conclusions and future work

To our knowledge, this work represents the first attempt to discover downward entailing operators. We introduced a unsupervised algorithm that is motivated by research in linguistics but employs simple distributional statistics in a novel fashion. Our algorithm is highly accurate and discovers many reasonable DE operators that are missing from pre-existing manually-built lists.

Since the algorithm is resource-lean — requiring no parser or tagger but only a list of NPIs — it can be immediately applied to languages where such lists exist, such as German and Romanian (Trawiński and Soehn, 2008). On the other hand, although the results are already quite good for English, it would be interesting to see what improvements could be gained by using more sophisticated syntactic information.

For languages where NPI lists are not extensive, one could envision applying an iterative co-learning approach: use the newly-derived DE operators to infer new NPIs, and then discover even more new DE operators given the new NPI list. (For English, our initial attempts at bootstrapping from our initial NPI list on the BLLIP corpus did not lead to substantially improved results.)

In practice, subcategorization is an important feature to capture. In Table 1, we indicate which subcategorizations are DE. An interesting extension of our work would be to try to automatically distinguish particular DE subcategorizations that are lexically apparent, e.g., '*innocent*' (not DE) vs. '*innocent of*' (as in '*innocent of burglary*', DE).

Our project provides a connection (among many) between the creation of textual entailment systems (the domain of language engineers) and the characterization of DE operators (the subject of study and debate among linguists). The prospect that our method might potentially eventually be refined in such a way so as to shed at least a little light on linguistic questions is a very appealing one, although we cannot be certain that any progress will be made on that front.

Acknowledgments We thank Roy Bar-Haim, Cleo Condoravdi, and Bill MacCartney for sharing their systems' lists and information about their work with us; Mats Rooth for helpful conversations; Alex Niculescu-Mizil for technical assistance; and Eugene Charniak for reassuring remarks. We also thank Marisa Ferrara Boston, Claire Cardie, Zhong Chen, Yejin Choi, Effi Georgala, Myle Ott, Stephen Purpura, and Ainur Yessenalina at Cornell University, the UT-Austin NLP group, Roy Bar-Haim, Bill MacCartney, and the anonymous reviewers for for their comments on this paper. This paper is based upon work supported in part by DHS grant N0014-07-1-0152, National Science Foundation grant No. BCS-0537606, a Yahoo! Research Alliance gift, a CU Provost's Award for Distinguished Scholarship, and a CU Institute for the Social Sciences Faculty Fellowship. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of any sponsoring institutions, the U.S. government, or any other entity.

References

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second PASCAL Recognising Textual Entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

Roy Bar-Haim, Jonathan Berant, Ido Dagan, Iddo Greental, Shachar Mirkin, Eyal Shnarch, and Idan Szpektor. Efficient semantic deduction and approximate matching over compact parse forests. In *Proceedings of TAC*, 2008.

Johan Bos and Katja Markert. Recognising textual entailment with robust logical inference. In Quiñonero Candela, Dagan, Magnini, and d'Alché Buc (2006), pages 404–426.

Christos Christodoulopoulos. Creating a natural logic inference system with combinatory categorial grammar. Master's thesis, University of Edinburgh, 2008.

Dick Crouch, Roser Saurí, and Abraham Fowler. AQUAINT pilot knowledge-based evaluation: Annotation guidelines. http://www2.parc.com/istl/groups/nltt/papers/aquaint_kb_pilot_evaluation_guide.pdf, 2005.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL Recognising Textual Entailment challenge. In Quiñonero Candela et al. (2006), pages 177–190.

David Dowty. The role of negative polarity and concord marking in natural language reasoning. In Mandy Harvey and Lynn Santelmann, editors, *Proceedings of SALT IV*, pages 114–144, Ithaca, New York, 1994. Cornell University.

Gilles Fauconnier. Polarity and the scale principle. In *Proceedings of the Chicago Linguistic Society (CLS)*, pages 188–199, 1975. Reprinted in Javier Gutierrez-Rexach (ed.), *Semantics: Critical Concepts in Linguistics*, 2003.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL Recognizing Textual Entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, 2007. URL http://www.aclweb.org/anthology/W/W07/W07-1401.

Anastasia Giannakidou. Licensing and sensitivity in polarity items: from downward entailment to nonveridicality. In *Proceedings of the Chicago Linguistic Society (CLS)*, 2002.

Jack Hoeksema. Monotonicity phenomena in natural language. *Linguistic Analysis*, 16:25–40, 1986.

Jack Hoeksema. As (of) yet. Appears in *Language and Cognition 3*, the 1992 yearbook of the research group for theoretical and experimental linguistics of the University of Groningen, 1993. http://www.let.rug.nl/hoeksema/asofyet.pdf.

Jack Hoeksema. Corpus study of negative polarity items. *IV-V Jornades de corpus linguistics 1996-1997*, 1997. http://odur.let.rug.nl/~hoeksema/docs/barcelona.html.

Wilfried Kürschner. *Studien zur Negation im Deutschen*. Narr, 1983.

William A. Ladusaw. *Polarity Sensitivity as Inherent Scope Relations*. Garland Press, New York, 1980. Ph.D. thesis date 1979.

John Lawler. Negation and NPIs. http://www.umich.edu/~jlawler/NPIs.pdf, 2005. Version of 10/29/2005.

Timm Lichte. Corpus-based acquisition of complex negative polarity items. In *ESSLLI Student Session*, 2005.

Timm Lichte and Jan-Philipp Soehn. The retrieval and classification of Negative Polarity Items using statistical profiles. In Sam Featherston and Wolfgang Sternefeld, editors, *Roots: Linguistics in Search of its Evidential Base*, pages 249–266. Mouton de Gruyter, 2007.

Marcia Linebarger. Negative polarity and grammatical representation. *Linguistics and philosophy*, 10:325–387, 1987.

Bill MacCartney and Christopher D. Manning. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 193–200, 2007.

Bill MacCartney and Christopher D. Manning. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 521–528, Manchester, UK, August 2008. Coling 2008 Organizing Committee. URL http://www.aclweb.org/anthology/C08-1066.

Bernardo Magnini. Slides for a presentation entitled "Semantic Knowledge for Textual Entailment". Symposium on Semantic Knowledge Discovery, Organization and Use, New York University, November 14 and 15, 2008.

Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. Computing relative polarity for textual inference. In *Proceedings of Inference in Computational Semantics (ICoS)*, 2006.

Polarity Items Bibliography. The polarity items bibliography. http://www.sfb441.uni-tuebingen.de/a5/pib/XML2HTML/list.html, 2008. Maintenance guaranteed only through December 2008.

Joaquin Quiñonero Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc, editors. *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, volume 3944 of *Lecture Notes in Computer Science (LNCS)*, 2006. Springer.

Víctor Sánchez Valencia. *Studies on natural logic and categorial grammar*. PhD thesis, University of Amsterdam, 1991.

Beata Trawiński and Jan-Philipp Soehn. A Multilingual Database of Polarity Items. In *Proceedings of LREC 2008, May 28–30*, Marrakech, Morocco, 2008.

Johan van Benthem. *Essays in Logical Semantics*. Reidel, Dordrecht, 1986.

Ton van der Wouden. *Negative contexts: Collocation, polarity and multiple negation*. Routledge, 1997.

Anke von Bergen and Karl von Bergen. *Negative Polarität im Englischen*. Gunter Narr, 1993. List extracted and compiled by Manfred Sailer, 2008, http://www.sfs.uni-tuebingen.de/~fr/esslli/08/byday/english-npi.pdf.

Kai von Fintel. NPI licensing, Strawson entailment, and context dependency. *Journal of Semantics*, 16:97–148, 1999.

The Role of Implicit Argumentation in Nominal SRL

Matt Gerber
Dept. of Computer Science
Michigan State University
gerberm2@msu.edu

Joyce Y. Chai
Dept. of Computer Science
Michigan State University
jchai@cse.msu.edu

Adam Meyers
Dept. of Computer Science
New York University
meyers@cs.nyu.edu

Abstract

Nominals frequently surface without overtly expressed arguments. In order to measure the potential benefit of nominal SRL for downstream processes, such nominals must be accounted for. In this paper, we show that a state-of-the-art nominal SRL system with an overall argument F_1 of 0.76 suffers a performance loss of more than 9% when nominals with implicit arguments are included in the evaluation. We then develop a system that takes implicit argumentation into account, improving overall performance by nearly 5%. Our results indicate that the degree of implicit argumentation varies widely across nominals, making automated detection of implicit argumentation an important step for nominal SRL.

1 Introduction

In the past few years, a number of studies have focused on verbal semantic role labeling (SRL). Driven by annotation resources such as FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005), many systems developed in these studies have achieved argument F_1 scores near 80% in large-scale evaluations such as the one reported by Carreras and Màrquez (2005).

More recently, the automatic identification of nominal argument structure has received increased attention due to the release of the NomBank corpus (Meyers, 2007a). NomBank annotates predicating nouns in the same way that PropBank annotates predicating verbs. Consider the following example of the verbal predicate *distribute* from the PropBank corpus:

(1) Freeport-McMoRan Energy Partners will be liquidated and [$_{Arg1}$ shares of the new company] [$_{Predicate}$ distributed] [$_{Arg2}$ to the partnership's unitholders].

The NomBank corpus contains a similar instance of the deverbal nominalization *distribution*:

(2) Searle will give [$_{Arg0}$ pharmacists] [$_{Arg1}$ brochures] [$_{Arg1}$ on the use of prescription drugs] for [$_{Predicate}$ distribution] [$_{Location}$ in their stores].

This instance demonstrates the annotation of split arguments (Arg1) and modifying adjuncts (Location), which are also annotated in PropBank. In cases where a nominal has a verbal counterpart, the interpretation of argument positions Arg0-Arg5 is consistent between the two corpora.

In addition to deverbal (i.e., event-based) nominalizations, NomBank annotates a wide variety of nouns that are not derived from verbs and do not denote events. An example is given below of the partitive noun *percent*:

(3) Hallwood owns about 11 [$_{Predicate}$ %] [$_{Arg1}$ of Integra].

In this case, the noun phrase headed by the predicate *%* (i.e., "about 11% of Integra") denotes a fractional part of the argument in position Arg1.

Since NomBank's release, a number of studies have applied verbal SRL techniques to the task of nominal SRL. For example, Liu and Ng (2007) reported an argument F_1 of 0.7283. Although this result is encouraging, it does not take into account nominals that surface without overt arguments. Consider the following example:

(4) The [$_{Predicate}$ distribution] represents [$_{NP}$ available cash flow] [$_{PP}$ from the partnership] [$_{PP}$ between Aug. 1 and Oct. 31].

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 146–154,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

As in (2), *distribution* in (4) has a noun phrase and multiple prepositional phrases in its environment, but not one of these constituents is an argument to *distribution* in (4); rather, any arguments are implicitly supplied by the surrounding discourse. As described by Meyers (2007a), instances such as (2) are called "markable" because they contain overt arguments, and instances such as (4) are called "unmarkable" because they do not. In the NomBank corpus, only markable instances have been annotated.

Previous evaluations (e.g., those by Jiang and Ng (2006) and Liu and Ng (2007)) have been based on markable instances, which constitute 57% of all instances of nominals from the NomBank lexicon. In order to use nominal SRL systems for downstream processing, it is important to develop and evaluate techniques that can handle markable as well as unmarkable nominal instances. To address this issue, we investigate the role of implicit argumentation for nominal SRL. This is, in part, inspired by the recent CoNLL Shared Task (Surdeanu et al., 2008), which was the first evaluation of syntactic and semantic dependency parsing to include unmarkable nominals. In this paper, we extend this task to constituent parsing with techniques and evaluations that focus specifically on implicit argumentation in nominals.

We first present our NomBank SRL system, which improves the best reported argument F_1 score in the markable-only evaluation from 0.7283 to 0.7630 using a single-stage classification approach. We show that this system, when applied to all nominal instances, achieves an argument F_1 score of only 0.6895, a loss of more than 9%. We then present a model of implicit argumentation that reduces this loss by 46%, resulting in an F_1 score of 0.7235 on the more complete evaluation task. In our analyses, we find that SRL performance varies widely among specific classes of nominals, suggesting interesting directions for future work.

2 Related work

Nominal SRL is related to nominal relation interpretation as evaluated in SemEval (Girju et al., 2007). Both tasks identify semantic relations between a head noun and other constituents; however, the tasks focus on different relations. Nominal SRL focuses primarily on relations that hold between nominalizations and their arguments, whereas the SemEval task focuses on a range of semantic relations, many of which are not applicable to nominal argument structure.

Early work in identifying the argument structure of deverbal nominalizations was primarily rule-based, using rule sets to associate syntactic constituents with semantic roles (Dahl et al., 1987; Hull and Gomez, 1996; Meyers et al., 1998). Lapata (2000) developed a statistical model to classify modifiers of deverbal nouns as underlying subjects or underlying objects, where subject and object denote the grammatical position of the modifier when linked to a verb.

FrameNet and NomBank have facilitated machine learning approaches to nominal argument structure. Gildea and Jurafsky (2002) presented an early FrameNet-based SRL system that targeted both verbal and nominal predicates. Jiang and Ng (2006) and Liu and Ng (2007) have tested the hypothesis that methodologies and representations used in PropBank SRL (Pradhan et al., 2005) can be ported to the task of NomBank SRL. These studies report argument F_1 scores of 0.6914 and 0.7283, respectively. Both studies also investigated the use of features specific to the task of NomBank SRL, but observed only marginal performance gains.

NomBank argument structure has also been used in the recent CoNLL Shared Task on Joint Parsing of Syntactic and Semantic Dependencies (Surdeanu et al., 2008). In this task, systems were required to identify syntactic dependencies, verbal and nominal predicates, and semantic dependencies (i.e., arguments) for the predicates. For nominals, the best semantic F_1 score was 0.7664 (Surdeanu et al., 2008); however this score is not directly comparable to the NomBank SRL results of Liu and Ng (2007) or the results in this paper due to a focus on different aspects of the problem (see the end of section 5.2 for details).

3 NomBank SRL

Given a nominal predicate, an SRL system attempts to assign surrounding spans of text to one of 23 classes representing core arguments, adjunct arguments, and the *null* or non-argument. Similarly to

verbal SRL, this task is traditionally formulated as a two-stage classification problem over nodes in the syntactic parse tree of the sentence containing the predicate.[1] In the first stage, each parse tree node is assigned a binary label indicating whether or not it is an argument. In the second stage, argument nodes are assigned one of the 22 non-null argument types. Spans of text subsumed by labeled parse tree nodes constitute arguments of the predication.

3.1 An improved NomBank SRL baseline

To investigate the effects of implicit argumentation, we first developed a system based on previous markable-only approaches. Our system follows many of the traditions above, but differs in the following ways. First, we replace the standard two-stage pipeline with a single-stage logistic regression model[2] that predicts arguments directly. Second, we model incorporated arguments (i.e., predicates that are also arguments) with a simple maximum likelihood model that predicts the most likely argument label for a predicate based on counts from the training data. Third, we use the following heuristics to resolve argument conflicts: (1) If two arguments overlap, the one with the higher probability is kept. (2) If two non-overlapping arguments are of the same type, the one with the higher probability is kept unless the two nodes are siblings, in which case both are kept. Heuristic (2) accounts for split argument constructions.

Our NomBank SRL system uses features that are selected with a greedy forward search strategy similar to the one used by Jiang and Ng (2006). The top half of Table 2 (next page) lists the selected argument features.[3] We extracted training nodes from sections 2-21 of NomBank, used section 24 for development and section 23 for testing. All parse trees were generated by Charniak's re-ranking syntactic parser (Charniak and Johnson, 2005). Following the evaluation methodology used by Jiang and Ng (2006) and Liu and Ng (2007), we obtained significantly better results, as shown in Table 1 above.[4]

	Dev. F_1	Testing F_1
Jiang and Ng (2006)	0.6677	0.6914
Liu and Ng (2007)	-	0.7283
This paper	0.7454	**0.7630**

Table 1: Markable-only NomBank SRL results for argument prediction using automatically generated parse trees. The f-measure statistics were calculated by aggregating predictions across all classes. "-" indicates that the result was not reported.

	Markable-only	All-token	% loss
P	0.7955	0.6577	-17.32
R	0.7330	0.7247	-1.13
F_1	0.7630	0.6895	-9.63

Table 3: Comparison of the markable-only and all-token evaluations of the baseline argument model.

3.2 The effect of implicit nominal arguments

The presence of implicit nominal arguments presents challenges that are not taken into account by the evaluation described above. To assess the impact of implicit arguments, we evaluated our NomBank SRL system over each token in the testing section. The system attempts argument identification for all singular and plural nouns that have at least one annotated instance in the training portion of the NomBank corpus (morphological variations included).

Table 3 gives a comparison of the results from the markable-only and all-token evaluations. As can be seen, assuming that all known nouns take overt arguments results in a significant performance loss. This loss is due primarily to a drop in precision caused by false positive argument predictions made for nominals with implicit arguments.

4 Accounting for implicit arguments in nominal SRL

A natural solution to the problem described above is to first distinguish nominals that bear overt arguments from those that do not. We treat this

[1] The syntactic parse can be based on ground-truth annotation or derived automatically, depending on the evaluation.

[2] We use LibLinear (Fan et al., 2008).

[3] For features requiring the identification of support verbs, we use the annotations provided in NomBank. Preliminary experiments show a small loss when using automatic support verb identification.

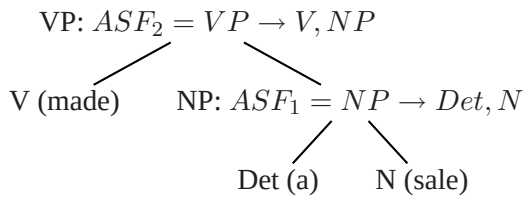
[4] As noted by Carreras and Màrquez (2005), the discrepancy between the development and testing results is likely due to poorer syntactic parsing performance on the development section.

	#	Description	N	S
Argument features	1	12 **&** parse tree path from n to $pred$		
	2	Position of n relative to $pred$ **&** parse tree path from n to $pred$	*	
	3	First word subsumed by n		
	4	12 **&** position of n relative to $pred$		
	5	12 **&** 14		
	6	Head word of n's parent		*
	7	Last word subsumed n		
	8	n's syntactic category **&** length of parse tree path from n to $pred$		
	9	First word of n's right sibling	*	*
	10	Production rule that expands the parent of $pred$		
	11	Head word of the right-most NP in n if n is a PP	*	
	12	Stem of $pred$		
	13	Parse tree path from n to the lowest common ancestor of n and $pred$		
	14	Head word of n		
	15	12 **&** n's syntactic category		
	16	Production rule that expands n's parent	*	*
	17	Parse tree path from n to the nearest support verb	*	
	18	Last part of speech (POS) subsumed by n	*	
	19	Production rule that expands n's left sibling		*
	20	Head word of n, if the parent of n is a PP		
	21	The POS of the head word of the right-most NP under n if n is a PP		
	...	Features 22-31 are available upon request	0	3
Nominal features	1	n's ancestor subcategorization frames (ASF) (see section 4)	*	
	2	n's word		
	3	Syntactic category of n's right sibling		
	4	Parse tree paths from n to each support verb	*	
	5	Last word of n's left sibling	*	*
	6	Parse tree path from n to previous nominal, with lexicalized source (see section 4)	*	
	7	Last word of n's right sibling	*	
	8	Production rule that expands n's left sibling	*	*
	9	Syntactic category of n		*
	10	PropBank markability score (see section 4)	*	
	11	Parse tree path from n to previous nominal, with lexicalized source and destination	*	
	12	Whether or not n is followed by PP	*	
	13	Parse tree path from n to previous nominal, with lexicalized destination	*	
	14	Head word of n's parent		*
	15	Whether or not n surfaces before a passive verb	*	*
	16	First word of n's left sibling	*	
	17	Parse tree path from n to closest support verb, with lexicalized destination	*	
	18	Whether or not n is a head	*	
	19	Head word of n's right sibling		
	20	Production rule that expands n's parent	*	*
	21	Parse tree paths from n to all support verbs, with lexicalized destinations	*	
	22	First word of n's right sibling	*	*
	23	Head word of n's left sibling	*	
	24	If n is followed by a PP, the head of that PP's object	*	
	25	Parse tree path from n to previous nominal	*	
	26	Token distance from n to previous nominal	*	
	27	Production rule that expands n's grandparent	*	

Table 2: Features, sorted by gain in selection algorithm. **&** denotes concatenation. The last two columns indicate (N)ew features (not used in Liu and Ng (2007)) and features (S)hared by the argument and nominal models.

as a binary classification task over token nodes. Once a nominal has been identified as bearing overt arguments, it is processed with the argument identification model developed in the previous section. To classify nominals, we use the features shown in the bottom half of Table 2, which were selected with the same algorithm used for the argument classification model. As shown by Table 2, the sets of features selected for argument and nominal classification are quite different, and many of the features used for nominal classification have not been previously used. Below, we briefly explain a few of these features.

Ancestor subcategorization frames (ASF) As shown in Table 2, the most informative feature is ASF. For a given token t, ASF is actually a set of sub-features, one for each parse tree node above t. Each sub-feature is indexed (i.e., named) by its distance from t. The value of an ASF sub-feature is the production rule that expands the corresponding node in the tree. An ASF feature with two sub-features is depicted below for the token "sale":



Parse tree path lexicalization A lexicalized parse tree path is one in which surface tokens from the beginning or end of the path are included in the path. This is a finer-grained version of the traditional parse tree path that captures the joint behavior of the path and the tokens it connects. For example, in the tree above, the path from "sale" to "made" with a lexicalized source and destination would be $sale : N \uparrow NP \uparrow VP \downarrow V : made$. Lexicalization increases sparsity; however, it is often preferred by the feature selection algorithm, as shown in the bottom half of Table 2.

PropBank markability score This feature is the probability that the context ($\pm$ 5 words) of a deverbal nominal is generated by a unigram language model trained over the PropBank argument words for the corresponding verb. Entities are normalized to their entity type using BBN's IdentiFinder, and adverbs are normalized to their related adjective using the *ADJADV* dictionary provided by NomBank. The normalization of adverbs is motivated by the fact that adverbial modifiers of verbs typically have a corresponding adjectival modifier for deverbal nominals.

	Precision	Recall	F_1
Baseline	0.5555	0.9784	0.7086
MLE	0.6902	0.8903	0.7776
LibLinear	0.8989	0.8927	0.8958

Table 4: Evaluation results for identifying nominals with explicit arguments.

5 Evaluation results

Our evaluation methodology reflects a practical scenario in which the nominal SRL system must process each token in a sentence. The system cannot safely assume that each token bears overt arguments; rather, this decision must be made automatically. In section 5.1, we present results for the automatic identification of nominals with overt arguments. Then, in section 5.2, we present results for the combined task in which nominal classification is followed by argument identification.

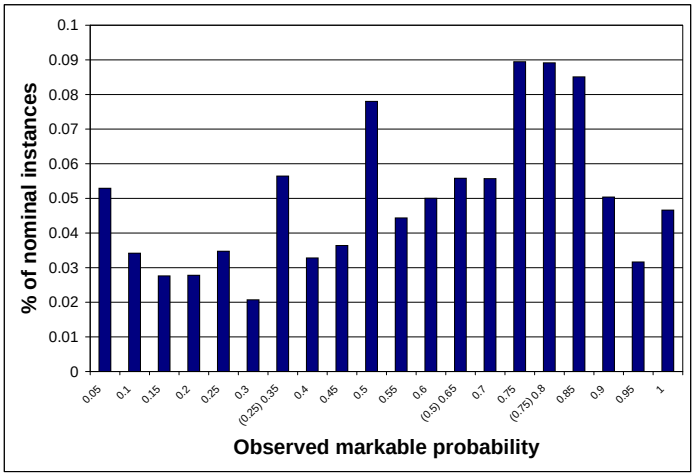
5.1 Nominal classification

Following standard practice, we train the nominal classifier over NomBank sections 2-21 using LibLinear and automatically generated syntactic parse trees. The prediction threshold is set to the value that maximizes the nominal F_1 score on development section (24), and the resulting model is tested over section 23. For comparison, we implemented the following simple classifiers.

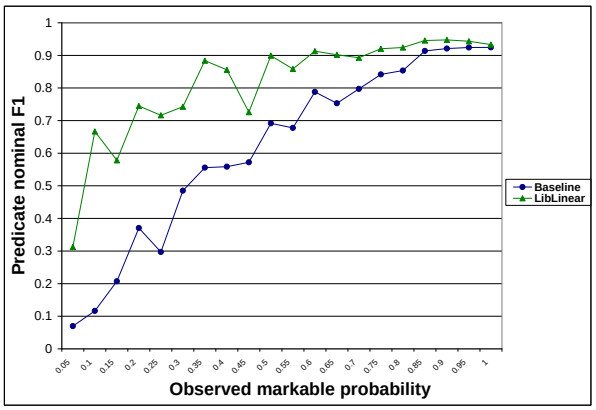
Baseline nominal classifier Classifies a token as overtly bearing arguments if it is a singular or plural noun that is markable in the training data. As shown in Table 4, this classifier achieves nearly perfect recall.[5]

MLE nominal classifier Operates similarly to

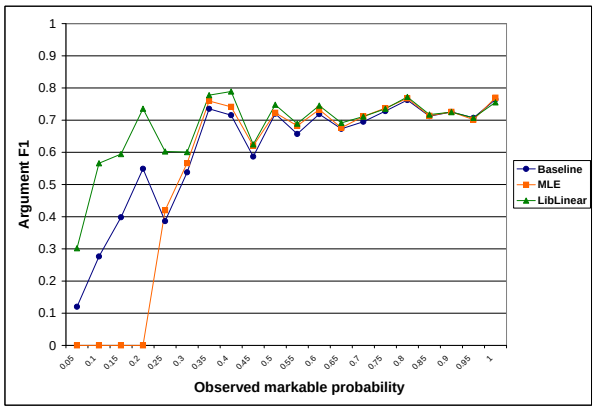
[5]Recall is less than 100% due to (1) part-of-speech errors from the syntactic parser and (2) nominals that were not annotated in the training data but exist in the testing data.



(a) Distribution of nominals. Each interval on the x-axis denotes a set of nominals that are markable between $(x-5)\%$ and $x\%$ of the time in the training data. The y-axis denotes the percentage of all nominal instances in TreeBank that is occupied by nominals in the interval. Quartiles are marked below the intervals. For example, quartile 0.25 indicates that one quarter of all nominal instances are markable 35% of the time or less.



(b) Nominal classification performance with respect to the distribution in Figure 1a. The y-axis denotes the combined F_1 for nominals in the interval.



(c) All-token argument classification performance with respect to the distribution in Figure 1a. The y-axis denotes the combined F_1 for nominals in the interval.

Figure 1: Evaluation results with respect to the distribution of nominals in TreeBank.

the baseline classifier, but also produces a score for the classification. The value of the score is equal to the probability that the nominal bears overt arguments, as observed in the training data. A prediction threshold is imposed on this score as determined by the development data ($t = 0.23$). As shown by Table 4, this exchanges recall for precision and leads to a significant increase in the overall F_1 score.

The last row in Table 4 shows the results for the LibLinear nominal classifier, which significantly outperforms the others, achieving balanced precision and recall scores near 0.9. In addition, it is able to recover from part-of-speech errors because it does not filter out non-noun instances; rather, it combines part-of-speech information with other lexical and syntactic features to classify nominals.

Interesting observations can be made by grouping nominals according to the probability with which they are markable in the corpus. Figure 1a gives the overall distribution of markable nominals in the training data. As shown, 50% of nominal instances are markable only 65% of the time or less, making nominal classification an important first step. Using this view of the data, Figure 1b presents the overall F_1 scores for the baseline and LibLinear nominal

classifiers.[6] As expected, gains in nominal classification diminish as nominals become more overtly associated with arguments. Furthermore, nominals that are rarely markable (i.e., those in interval 0.05) remain problematic due to a lack of positive training instances and the unbalanced nature of the classification task.

5.2 Combined nominal-argument classification

We now turn to the task of combined nominal-argument classification. In this task, systems must first identify nominals that bear overt arguments. We evaluated three configurations based on the nominal classifiers from the previous section. Each configuration uses the argument classification model from section 3.

As shown in Table 3, overall argument classification F_1 suffers a loss of more than 9% under the assumption that all known nouns bear overt arguments. This corresponds precisely to using the baseline nominal classifier in the combined nominal-argument task. The MLE nominal classifier is able to reduce this loss by 25% to an F_1 of 0.7080. The LibLinear nominal classifier reduces this loss by 46%, resulting in an overall argument classification F_1 of 0.7235. This improvement is the direct result of filtering out nominal instances that do not bear overt arguments.

Similarly to the nominal evaluation, we can view argument classification performance with respect to the probability that a nominal bears overt arguments. This is shown in Figure 1c for the three configurations. The configuration using the MLE nominal classifier obtains an argument F_1 of zero for nominals below its prediction threshold. Compared to the baseline nominal classifier, the LibLinear classifier achieves argument classification gains as large as 150.94% (interval 0.05), with an average gain of 52.87% for intervals 0.05 to 0.4. As with nominal classification, argument classification gains diminish for nominals that express arguments more overtly - we observe an average gain of only 2.15% for intervals 0.45 to 1.00. One possible explanation for this is that the argument prediction model has substantially more training data for the nominals in intervals 0.45 to 1.00. Thus, even if the nominal classifier makes a false positive prediction in the 0.45 to 1.00 interval range, the argument model may correctly avoid labeling any arguments.

[6] Baseline and MLE are identical above the MLE threshold.

	Nominals		
	Deverbal	Deverbal-like	Other
Baseline	0.7975	0.6789	0.6757
MLE	0.8298	0.7332	0.7486
LibLinear	0.9261	0.8826	0.8905
	Arguments		
Baseline	0.7059	0.6738	0.7454
MLE	0.7206	0.6641	0.7675
LibLinear	0.7282	0.7178	0.7847

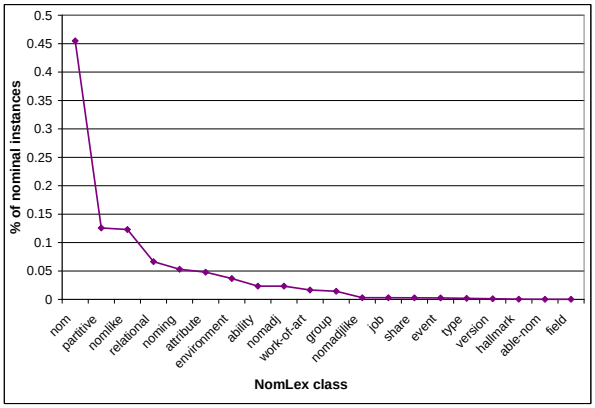
Table 5: Nominal and argument F_1 scores for deverbal, deverbal-like, and other nominals in the all-token evaluation.

As noted in section 2, these results are not directly comparable to the results of the recent CoNLL Shared Task (Surdeanu et al., 2008). This is due to the fact that the semantic labeled F_1 in the Shared Task combines predicate and argument predictions into a single score. The same combined F_1 score for our best two-stage nominal SRL system (logistic regression nominal and argument models) is 0.7806; however, this result is not precisely comparable because we do not identify the predicate role set as required by the CoNLL Shared Task.

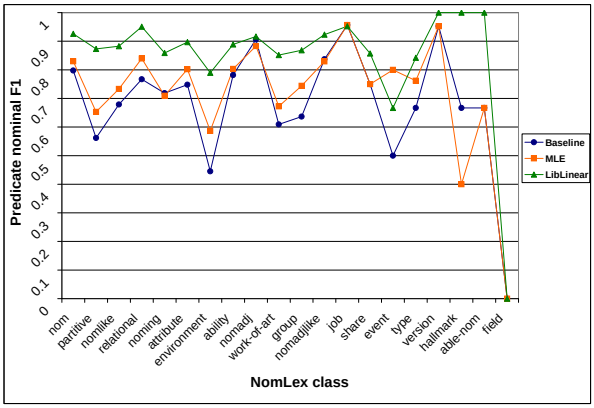
5.3 NomLex-based analysis of results

As demonstrated in section 1, NomBank annotates many classes of deverbal and non-deverbal nominals, which have been categorized on syntactic and semantic bases in NomLex-PLUS (Meyers, 2007b). To help understand what types of nominals are particularly affected by implicit argumentation, we further analyzed performance with respect to these classes.

Figure 2a shows the distribution of nominals across classes defined by the NomLex resource. As shown in Figure 2b, many of the most frequent classes exhibit significant gains. For example, the classification of partitive nominals (13% of all nominal instances) with the LibLinear classifier results in gains of 55.45% and 33.72% over the baseline and MLE classifiers, respectively. For the 5 most common classes, which constitute 82% of all nominals instances, we observe average gains of 27.47% and 19.30% over the baseline and MLE classifiers,



(a) Distribution of nominals across the NomLex classes. The y-axis denotes the percentage of all nominal instances that is occupied by nominals in the class.



(b) Nominal classification performance with respect to the NomLex classes in Figure 2a. The y-axis denotes the combined F_1 for nominals in the class.

Figure 2: Evaluation results with respect to NomLex classes.

respectively.

Table 5 separates nominal and argument classification results into sets of deverbal (NomLex class *nom*), deverbal-like (NomLex class *nom-like*), and all other nominalizations. A deverbal-like nominal is closely related to some verb, although not morphologically. For example, the noun *accolade* shares argument interpretation with *award*, but the two are not morphologically related. As shown by Table 5, nominal classification tends to be easier - and argument classification harder - for deverbals when compared to other types of nominals. The difference in argument F_1 between deverbal/deverbal-like nominals and the others is due primarily to relational nominals, which are relatively easy to classify (Figure 2b); additionally, relational nominals exhibit a high rate of argument incorporation, which is easily handled by the maximum-likelihood model described in section 3.1.

6 Conclusions and future work

The application of nominal SRL to practical NLP problems requires a system that is able to accurately process each token it encounters. Previously, it was unclear whether the models proposed by Jiang and Ng (2006) and Liu and Ng (2007) would operate effectively in such an environment. The systems described by Surdeanu et al. (2008) are designed with this environment in mind, but their evaluation did not focus on the issue of implicit argumentation. These two problems motivate the work presented in this paper.

Our contribution is three-fold. First, we improve upon previous nominal SRL results using a single-stage classifier with additional new features. Second, we show that this model suffers a substantial performance degradation when evaluated over nominals with implicit arguments. Finally, we identify a set of features - many of them new - that can be used to reliably detect nominals with explicit arguments, thus significantly increasing the performance of the nominal SRL system.

Our results also suggest interesting directions for future work. As described in section 5.2, many nominals do not have enough labeled training data to produce accurate argument models. The generalization procedures developed by Gordon and Swanson (2007) for PropBank SRL and Padó et al. (2008) for NomBank SRL might alleviate this problem. Additionally, instead of ignoring nominals with implicit arguments, we would prefer to identify the implicit arguments using information contained in the surrounding discourse. Such inferences would help connect entities and events across sentences, providing a fuller interpretation of the text.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful suggestions. The first two authors were supported by NSF grants IIS-0535112 and IIS-0347548, and the third author was supported by NSF grant IIS-0534700.

References

Collin Baker, Charles Fillmore, and John Lowe. 1998. The Berkeley FrameNet project. In Christian Boitet and Pete Whitelock, editors, *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 86–90, San Francisco, California. Morgan Kaufmann Publishers.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling.

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

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

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

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. Semeval-2007 task 04: Classification of semantic relations between nominals. In *Proceedings of the 4th International Workshop on Semantic Evaluations*.

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

Z. Jiang and H. Ng. 2006. Semantic role labeling of nombank: A maximum entropy approach. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*.

Maria Lapata. 2000. The automatic interpretation of nominalizations. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 716–721. AAAI Press / The MIT Press.

Chang Liu and Hwee Ng. 2007. Learning predictive structures for semantic role labeling of nombank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 208–215, Prague, Czech Republic, June. Association for Computational Linguistics.

Adam Meyers. 2007a. Annotation guidelines for nombank - noun argument structure for propbank. Technical report, New York University.

Adam Meyers. 2007b. Those other nombank dictionaries. Technical report, New York University.

Sebastian Padó, Marco Pennacchiotti, and Caroline Sporleder. 2008. Semantic role assignment for event nominalisations by leveraging verbal data. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 665–672, Manchester, UK, August. Coling 2008 Organizing Committee.

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

Sameer Pradhan, Wayne Ward, and James H. Martin. 2005. Towards robust semantic role labeling. In *Association for Computational Linguistics*.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England, August. Coling 2008 Organizing Committee.

Jointly Identifying Predicates, Arguments and Senses using Markov Logic

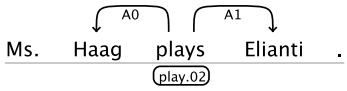
Ivan Meza-Ruiz[*] **Sebastian Riedel**[†‡]
[*]School of Informatics, University of Edinburgh, UK
[†]Department of Computer Science, University of Tokyo, Japan
[‡]Database Center for Life Science, Research Organization of Information and System, Japan
[*]I.V.Meza-Ruiz@sms.ed.ac.uk [†] sebastian.riedel@gmail.com

Abstract

In this paper we present a Markov Logic Network for Semantic Role Labelling that jointly performs predicate identification, frame disambiguation, argument identification and argument classification for all predicates in a sentence. Empirically we find that our approach is competitive: our best model would appear on par with the best entry in the CoNLL 2008 shared task open track, and at the 4th place of the closed track—right behind the systems that use significantly better parsers to generate their input features. Moreover, we observe that by fully capturing the complete SRL pipeline in a single probabilistic model we can achieve significant improvements over more isolated systems, in particular for out-of-domain data. Finally, we show that despite the joint approach, our system is still efficient.

1 Introduction

Semantic Role Labelling (SRL, Márquez et al., 2008) is generally understood as the task of identifying and classifying the semantic arguments and modifiers of the predicates mentioned in a sentence. For example, in the case of the following sentence:



we are to find out that for the predicate token "plays" with sense "play a role" (play.02) the phrase headed by the token "Haag" is referring to the player (A0) of the play event, and the phrase headed by the token "Elianti" is referring to the role (A1) being played. SRL is considered as a key task for applications that require to answer "Who", "What", "Where", etc. questions, such as Information Extraction, Question Answering and Summarization.

Any real-world SRL system needs to make several decisions, either explicitly or implicitly: which are the predicate tokens of a sentence (predicate identification), which are the tokens that have semantic roles with respect to these predicates (argument identification), which are the roles these tokens play (argument classification), and which is the sense of the predicate (sense disambiguation).

In this paper we use Markov Logic (ML), a Statistical Relational Learning framework that combines First Order Logic and Markov Networks, to develop a joint probabilistic model over all decisions mentioned above. The following paragraphs will motivate this choice.

First, it allows us to readily capture *global correlations* between decisions, such as the constraint that a predicate can only have one agent. This type of correlations has been successfully exploited in several previous SRL approaches (Toutanova et al., 2005; Punyakanok et al., 2005).

Second, we can use the joint model to evaluate the benefit of incorporating decisions into the joint model that either have not received much attention within the SRL community (predicate identification and sense disambiguation), or been largely made in isolation (argument identification and classification for all predicates of a sentence).

Third, our ML model is essentially a template that describes a class of Markov Networks. Algorithms can perform inference in terms of this template with-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 155–163,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

out ever having to fully instantiate the complete Markov Network (Riedel, 2008; Singla and Domingos, 2008). This can dramatically improve the efficiency of an SRL system when compared to a propositional approach such as Integer Linear Programming (ILP).

Finally, when it comes to actually building an SRL system with ML there are "only" four things to do: preparing input data files, converting output data files, and triggering learning and inference. The remaining work can be done by an off-the-shelf Markov Logic interpreter. This is to be contrasted with pipeline systems where several components need to be trained and connected, or Integer Linear Programming approaches for which we need to write additional wrapper code to generate ILPs.

Empirically we find that our system is competitive—our best model would appear on par with the best entry in the CoNLL 2008 shared task open track, and at the 4th place of the closed track—right behind systems that use significantly better parsers[1] to generate their input features.

We also observe that by integrating frame disambiguation into the joint SRL model, and by extracting all arguments for all predicates in a sentence simultaneously, significant improvements compared to more isolated systems can be achieved. These improvements are particularly large in the case of out-of-domain data, suggesting that a joint approach helps to increase the robustness of SRL. Finally, we show that despite the joint approach, our system is still efficient.

Our paper is organised as follows: we first introduce ML (section 2), then we present our model in terms of ML (section 3) and illustrate how to perform learning and inference with it (section 4). How this model will be evaluated is explained in section 5 with the corresponding evaluation presented in section 6. We conclude in section 7.

2 Markov Logic

Markov Logic (ML, Richardson and Domingos, 2005) is a Statistical Relational Learning language based on First Order Logic and Markov Networks. It can be seen as a formalism that extends First Order Logic to allow formulae that can be violated with some penalty. From an alternative point of view, it is an expressive template language that uses First Order Logic formulae to instantiate Markov Networks of repetitive structure.

Let us describe ML by considering the predicate identification task. In ML we can model this task by first introducing a set of logical predicates[2] such as *isPredicate(Token)* or *word(Token,Word)*. Then we specify a set of weighted first order formulae that define a distribution over sets of ground atoms of these predicates (or so-called *possible worlds*).

Ideally, the distribution we define with these weighted formulae assigns high probability to possible worlds where SRL predicates are correctly identified and a low probability to worlds where this is not the case. For example, a suitable set of weighted formulae would assign a high probability to the world[3]

$$\{word\,(1, \text{Haag})\,, word(2, \text{plays}),\\ word(3, \text{Elianti}), isPredicate(2)\}$$

and a low one to

$$\{word\,(1, \text{Haag})\,, word(2, \text{plays}),\\ word(3, \text{Elianti}), isPredicate(3)\}$$

In Markov Logic a set of weighted formulae is called a *Markov Logic Network* (MLN). Formally speaking, an MLN M is a set of pairs (ϕ, w) where ϕ is a first order formula and w a real weight. M assigns the probability

$$p\,(\mathbf{y}) = \frac{1}{Z} \exp\left(\sum_{(\phi,w)\in M} w \sum_{\mathbf{c}\in C^\phi} f^\phi_{\mathbf{c}}\,(\mathbf{y})\right) \quad (1)$$

to the possible world $\mathbf{y}$. Here C^ϕ is the set of all possible bindings of the free variables in ϕ with the constants of our domain. $f^\phi_{\mathbf{c}}$ is a feature function that returns 1 if in the possible world $\mathbf{y}$ the *ground formula* we get by replacing the free variables in ϕ by the constants in $\mathbf{c}$ is true and 0 otherwise. Z is a normalisation constant. Note that this distribution corresponds to a Markov Network (the so-called *Ground Markov Network*) where nodes represent ground atoms and factors represent ground formulae.

[1] Our unlabelled accuracy for syntactic dependencies is at least 3% points under theirs.

[2] In the cases were is not obvious whether we refer to SRL or ML predicates we add the prefix SRL or ML, respectively.

[3] "Haag plays Elianti" is a segment of a sentence in the training corpus.

For example, if M contains the formula ϕ

$$word\,(x, \text{take}) \Rightarrow isPredicate\,(x)$$

then its corresponding log-linear model has, among others, a feature f^{ϕ}_{t1} for which x in ϕ has been replaced by the constant t_1 and that returns 1 if

$$word\,(1, \text{take}) \Rightarrow isPredicate\,(1)$$

is true in $\mathbf{y}$ and 0 otherwise.

We will refer predicates such as *word* as *observed* because they are known in advance. In contrast, *isPredicate* is *hidden* because we need to infer it at test time.

3 Model

Conceptually we divide our SRL system into three stages: one stage that identifies the predicates of a sentence, one stage that identifies and classifies the arguments of these predicates, and a final stage that predicts the sense of each predicate. We should stress that this architecture is intended to illustrate a typical SRL system, and to describe the pipeline-based approach we will compare our models to. However, it does *not* correspond to the way inference is performed in our proposed model—we jointly infer *all* decisions described above.

Note that while the proposed division into conceptual stages seems somewhat intuitive, it is by no means uncontroversial. In fact, for the CoNLL 2008 shared task slightly more than one half of the participants performed sense disambiguation before argument identification and classification; most other participants framed the problem in the reverse order.[4]

We define five hidden predicates for the three stages of the task. Figure 1 illustrates these predicates and the stage they belong to. For predicate identification, we use the predicate *isPredicate*. *isPredicate(p)* indicates that the word in the position p is an SRL predicate. For argument identification and classification, we use the predicates *isArgument*, *hasRole* and *role*. The atom *isArgument(a)* signals that the word in the position a is a SRL argument of some (unspecified) SRL predicate while *hasRole(p,a)* indicates that the token at position a is

[4]However, for almost all pipeline based systems, predicate identification was the first stage of the role labelling process.

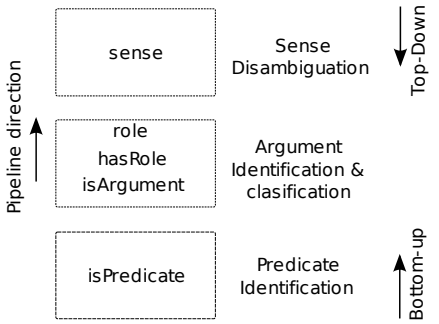


Figure 1: MLN hidden predicates divided in stages

an argument of the predicate in position p. The predicate *role(p,a,r)* corresponds to the decision that the argument at position a has the role r with respect to the predicate in position p. Finally, for sense disambiguation we define the predicate *sense(p,e)* which signals that the predicate in position p has the sense e.

Before we continue to describe the formulae of our Markov Logic Network we would like to highlight the introduction of the $isArgument$ predicate mentioned above. This predicate corresponds to a decision that is usually made implicitly: a token is an argument if there exists a predicate for which it plays a semantic role. Here we model this decision explicitly, assuming that there exist cases where a token clearly has to be an argument of some predicate, regardless of which predicate in the sentence this might be. It is this assumption that requires us to infer the arguments for all predicates of a sentence at once—otherwise we cannot make sure that for a marked argument there exists at least one predicate for which the argument plays a semantic role.

In addition to the hidden predicates, we define observable predicates to represent the information available in the corpus. Table 1 presents these predicates.

3.1 Local formulae

A formula is *local* if its groundings relate any number of observed ground atoms to exactly one hidden ground atom. For example, two groundings of the local formula

$$lemma(p, +l_1) \wedge lemma(a, +l_2) \Rightarrow hasRole(p, a)$$

can be seen in the Factor Graph of Figure 2. Both connect a single hidden *hasRole* ground atom with

word(i,w)	Token i has word w
lemma(i,l)	Token i has lemma l
ppos(i,p)	Token i has POS tag p
cpos(i,p)	Token i has coarse POS tag p
voice(i,v)	Token i is verb and has voice v (Active/Passive).
subcat(i,f)	Token i has subcategorization frame f
dep(i,j,d)	Token h is head of token m and has dependency label d
palmer(i,j)	Token j can be semantic argument for token i according to high recall heuristic*
depPath(i,j,p)	Dependency path between tokens i and j is p*
depFrame(i,j,f)	f is a syntactic (dependency) frame in which tokens i and j are designated as "pivots"*

Table 1: Observable predicates; predicates marked with * are dependency parsing-based versions for features of Xue and Palmer (2004).

two observed *lemma* ground atoms. The + notation indicates that the MLN contains one instance of the rule, with a separate weight, for each assignment of the variables with a plus sign (**?**).

The local formulae for *isPredicate*, *isArgument* and *sense* aim to capture the relation of the tokens with their lexical and syntactic surroundings. This includes formulae such as

$$subcat(p, +f) \Rightarrow isPredicate(p)$$

which implies that a certain token is a predicate with a weight that depends on the subcategorization frame of the token. Further local formulae are constructed using those observed predicates in table 1 that relate single tokens and their properties.

The local formulae for *role* and *hasRole* focus on properties of the predicate and argument token—the formula illustrated in figure 2 is an example of this—and on the relation between the two tokens. An example of the latter type is the formula

$$depPath(p, a, +d) \Rightarrow role(p, a, +r)$$

which implies that token a plays the semantic role r with respect to token p, and for which the weight depends on the syntactic (dependency) path d between p and a and on the actual role to assign. Again, further formulae are constructed using the observed

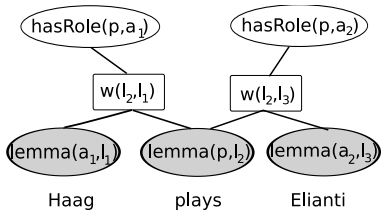


Figure 2: Factor graph for the first local formula in section 3.1. Here round nodes represent variables (corresponding to the states of ground atoms) and the rectangular nodes represent the factor and their parameters attached to the ground formulae.

predicates in table 1; however, this time we consider both predicates that relate tokens to their individual properties and predicates that describe the relation between tokens.

Unfortunately, the complete set of local formulae is too large to be exhaustively described in this paper. Its size results from the fact that we also consider conjunctions of several atoms as conditions, and lexical windows around tokens. Hence, instead of describing all local formulae we refer the reader to our MLN model files.[5] They can be used both as a reference and as input to our Markov Logic Engine,[6] and thus allow the reader to easily reproduce our results.

3.2 Global formulae

Global formulae relate several hidden ground atoms. We use this type of formula for two purposes: to ensure consistency between the predicates of all SRL stages, and to capture some of our background knowledge about SRL. We will refer to formulae that serve the first purpose as *structural constraints*.

For example, a structural constraint is given by the (deterministic) formula

$$role(p, a, r) \Rightarrow hasRole(p, a)$$

which ensures that, whenever the argument a is given a label r with respect to the predicate p, this argument must be an argument of a as denoted by *hasRole(p,a)*. Note that this formula by itself models the traditional "bottom-up" argument identification and classification pipeline (Xue and Palmer, 2004):

[5] http://code.google.com/p/thebeast/source/browse/#svn/mlns/naacl-hlt

[6] http://code.google.com/p/thebeast

it is possible to not assign a role r to an predicate-argument pair (p, a) proposed by the identification stage; however, it is impossible to assign a role r to token pairs (p, a) that have not been proposed as potential arguments.

An example of another class of structural constraints is

$$hasRole(p, a) \Rightarrow \exists r.role(p, a, r)$$

which, by itself, models an inverted or "top-down" pipeline. In this architecture the argument classification stage can assign roles to tokens that have not been proposed by the argument identification stage. However, it must assign a label to any token pair the previous stage proposes.

For the SRL predicates that perform a labelling task (*role* and *sense*) we also need a structural constraint which ensures that not more than one label is assigned. For instance,

$$(role(p, a, r_1) \wedge r_1 \neq r_2 \Rightarrow \neg role(p, a, r_2))$$

forbids two different semantic roles for a pair of words.

There are three global formulae that capture our linguistic background knowledge. The first one is a deterministic constraint that had been frequently applied in the SRL literature. It forbids cases where distinct arguments of a predicate have the same role unless the role describes a modifier:

$$role\,(p, a_1, r) \wedge \neg mod\,(r) \wedge a_1 \neq a_2 \Rightarrow \neg role\,(p, a_2, r)$$

The second "linguistic" global formula is

$$role(p, a, +r) \wedge lemma(p, +l) \Rightarrow sense(p, +s)$$

which implies that when a predicate p with lemma l has an argument a with role r it has to have the sense s. Here the weight depends on the combination of role r, lemma l and sense s.

The third and final "linguistic" global formula is

$$lemma(p, +l) \wedge ppos(a, +p) \wedge hasRole(p, a) \Rightarrow sense(p, +f)$$

It implies that if a predicate p has the lemma l and an argument a with POS tag p it has to have the sense s. This time the weight depends on the combination of POS tag p, lemma l and sense s.

Note that the final two formulae evaluate the semantic frame of a predicate and become local formulae in a pipeline system that performs sense disambiguation after argument identification and classification.

Table 2 summarises the global formulae we use in this work.

4 Inference and Learning

Assuming that we have an MLN, a set of weights and a given sentence then we need to predict the choice of predicates, frame types, arguments and role labels with maximal *a posteriori* probability (MAP). To this end we apply a method that is both exact and efficient: Cutting Plane Inference (CPI, Riedel, 2008) with Integer Linear Programming (ILP) as *base solver*.

Instead of fully instantiating the Markov Network that a Markov Logic Network describes, CPI begins with a subset of factors/edges—in our case we use the factors that correspond to the local formulae of our model—and solves the MAP problem for this subset using the base solver. It then inspects the solution for ground formulae/features that are not yet included but could, if added, lead to a different solution—this process is usually referred to as *separation*. The ground formulae that we have found are added and the network is solved again. This process is repeated until the network does not change anymore.

This type of algorithm could also be realised for an ILP formulation of SRL. However, it would require us to write a dedicated separation routine for each type of constraint we want to add. In Markov Logic, on the other hand, separation can be generically implemented as the search for variable bindings that render a weighted first order formulae true (if its weight is negative) or false (if its weight is positive). In practise this means that we can try new global formulae/constraints without any additional implementation overhead.

We learn the weights associated with each MLN using 1-best MIRA (Crammer and Singer, 2003) Online Learning method. As MAP inference method that is applied in the inner loop of the online learner we apply CPI, again with ILP as base

Bottom-up	$sense(p, s) \Rightarrow isPredicate(p)$ $hasRole(p, a) \Rightarrow isPredicate(p)$ $hasRole(p, a) \Rightarrow isArgument(a)$ $role(p, a, r) \Rightarrow hasLabel(p, a)$
Top-Down	$isPredicate(p) \Rightarrow \exists s.sense(p, s)$ $isPredicate(p) \Rightarrow \exists a.hasRole(p, a)$ $isArgument(a) \Rightarrow \exists p.hasRole(p, a)$ $hasLabel(p, a) \Rightarrow \exists r.role(p, a, r)$
Unique Labels	$role(p, a, r_1) \wedge r_1 \neq r_2 \Rightarrow \neg role(p, a, r_2)$ $sense(p, s_1) \wedge s_1 \neq s_2 \Rightarrow \neg sense(p, r_2)$
Linguistic	$role\,(p, a_1, r) \wedge \neg mod\,(r) \wedge a_1 \neq a_2 \Rightarrow \neg role\,(p, a_2, r)$ $lemma(p, +l) \wedge ppos(a, +p) \wedge hasRole(p, a) \Rightarrow sense(p, +f)$ $lemma(p, +l) \wedge role(p, a, +r) \Rightarrow sense(p, +f)$

Table 2: Global formulae for ML model

solver.

5 Experimental Setup

For training and testing our SRL systems we used a version of the CoNLL 2008 shared task (Surdeanu et al., 2008) dataset that only mentions verbal predicates, disregarding the nominal predicates available in the original corpus.[7] While the original (open track) corpus came with MALT (Nivre et al., 2007) dependencies, we observed slightly better results when using the dependency parses generated with a Charniak parser (Charniak, 2000). Hence we used the latter for all our experiments.

To assess the performance of our model, and it to evaluate the possible gains to be made from considering a joint model of the complete SRL pipeline, we set up several systems. The *full* system uses a Markov Logic Network with all local and global formulae described in section 3. For the *bottom-up* system we removed the structural top-down constraints from the complete model—previous work Riedel and Meza-Ruiz (2008) has shown that this can lead to improved performance. The *bottom-up (-arg)* system is equivalent to the bottom-up system, but it does not include any formulae that mention the hidden $isArgument$ predicate.

For the systems presented so far we perform joint inference and learning. The *pipeline* system differs in this regard. For this system we train a separate model for each stage in the pipeline of figure 1. The predicate identification stage identifies the predicates (using all local $isPredicate$ formulae) of a sentence. The next stage predicts arguments and their roles for the identified predicates. Here we include all local and global formulae that involve only the predicates of this stage. In the last stage we predict the sense of each identified predicate using all formulae that involve the $sense$, without the structural constraints that connect the $sense$ predicate to the previous stages of the pipeline (these constraints are enforced by architecture).

6 Results

Table 3 shows the results of our systems for the CoNLL 2008 development set and the WSJ and brown test sets. The scores are calculated using the semantic evaluation metric of the CoNLL-08 shared task (Surdeanu et al., 2008). This metric measures the precision, recall and F_1 score of the recovered semantic dependencies. A semantic dependency is created for each predicate and its arguments, the label of such dependency is the role of the argument. Additionally, there is a semantic dependency for each predicate and a $ROOT$ argument which has the sense of the predicate as label.

To put these results into context, let us compare them to those of the participants of the CoNLL 2008 shared task (see the last three rows of table 3).[8] Our best model, Bottom-up, would reach the highest F_1 WSJ score, and second highest Brown score, for the open track. Here the best-performing participant was Vickrey and Koller (2008).

Table 3 also shows the results of the best (Johansson and Nugues, 2008) and fourth best sys-

[7] The reason for this choice where license problems.

[8] Results of other systems were extracted from Table 16 of the shared task overview paper (Surdeanu et al., 2008).

tem (Zhao and Kit, 2008) of the closed track. We note that we do significantly worse than Johansson and Nugues (2008), and roughly equivalent to Zhao and Kit (2008); this places us on the fourth rank of 19 participants. However, note that all three systems above us, as well as Zhao and Kit (2008), use parsers with at least about 90% (unlabelled) accuracy on the WSJ test set (Johansson's parser has about 92% unlabelled accuracy).[9] By contrast, with about 87% unlabelled accuracy our parses are significantly worse.

Finally, akin to Riedel and Meza-Ruiz (2008) we observe that the bottom-up joint model performs better than the full joint model.

System	Devel	WSJ	Brown
Full	76.93	79.09	67.64
Bottom-up	77.96	80.16	68.02
Bottom-up (-arg)	77.57	79.37	66.70
Pipeline	75.69	78.19	64.66
Vickrey	N/A	79.75	69.57
Johansson	N/A	86.37	71.87
Zhao	N/A	79.40	66.38

Table 3: Semantic F_1 scores for our systems and three CoNLL 2008 shared task participants. The Bottom-up results are statistically significantly different to all others (i.e., $\rho \leq 0.05$ according to the sign test).

6.1 Joint Model vs. Pipeline

Table 3 suggests that by including sense disambiguation into the joint model (as is the case for all systems but the pipeline) significant improvements can be gained. Where do these improvements come from? We tried to answer this question by taking a closer look at how accurately the pipeline predicts the $isPredicate$, $isArgument$, $hasRole$, $role$ and $sense$ relations, and how this compares to the result of the joint full model.

Table 4 shows that the joint model mainly does better when it comes to predicting the right predicate senses. This is particularly true for the case of the Brown corpus—here we gain about 10% points. These results suggest that a more joint approach may be particularly useful in order to increase the robustness of an SRL system in out-of-domain scenarios.[10]

[9] Since our parses use a different label set we could not compare labelled accuracy.

	WSJ		Brown	
	Pipe.	Fu.	Pipe.	Fu.
isPredicate	96.6	96.5	92.2	92.5
isArgument	90.3	90.6	85.9	86.9
hasRole	88.0	87.9	83.6	83.8
role	75.4	75.5	64.2	64.6
sense	85.5	88.5	67.3	77.1

Table 4: F_1 scores for M predicates; Pipe. refers to the Pipeline system, Fu. to the full system.

6.2 Modelling if a Token is an Argument

In table 3 we also observe that improvements can be made if we explicitly model the decision whether a token is a semantic argument of some predicate or not. As we mentioned in section 3, this aspect of our model requires us to jointly perform inference for all predicates of a sentence, and hence our results justify the per-sentence SRL approach proposed in this paper.

In order to analyse where these improvements come from, we again list our results on a per-SRL-predicate basis. Table 5 shows that by including the $isArgument$ predicate and the corresponding formulae we gain around 0.6% and 1.0% points across the board for WSJ and Brown, respectively.[11] As shown in table 3, these improvements result in about 1.0% improvements for both WSJ and Brown in terms of the CoNLL 2008 metric. Hence, an explicit model of the "is an argument" decision helps the SRL at all levels.

How the $isArgument$ helps to improve the overall role labelling score can be illustrated with the example in figure 3. Here the model without a hidden $isArgument$ predicate fails to attach the preposition "on" to the predicate "start.01" (here 01 refers to the sense of the predicate). Apparently the model has not enough confidence to assign the preposition to either "start.01" or "get.03", so it just drops the argument altogether. However, because the $isArgument$ model knows that most prepositions have to be modifying some predicate, pres-

[10] The differences between results of the full and joint model are statistically significant with the exception of the results for the $isPredicate$ predicate for the WSJ test set.

[11] The differences between results of the w/ and w/o model are statistically significant with the exception of the results for the $sense$ predicate for the Brown test set.

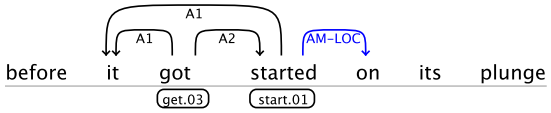


Figure 3: Segment of the CoNLL 2008 development set for which the bottom-up model w/o $isArgument$ predicate fails to attach the preposition "on" as an "AM-LOC" for "started". The joint bottom-up model attaches the preposition correctly.

sure is created that forces a decision between the two predicates. And because for the role model "start.01" looks like a better fit than "get.03", the correct attachment is found.

	WSJ		Brown	
	w/o	w/	w/o	w/
isPredicate	96.3	96.5	91.4	92.5
hasRole	87.1	87.7	82.5	83.6
role	76.9	77.5	65.2	66.2
sense	88.3	89.0	76.1	77.5

Table 5: F_1 scores for ML predicates; w/o refers to a Bottom-up system without isArgument predicate, w/ refers to a Bottom-up system with isArgument predicate.

6.3 Efficiency

In the previous sections we have shown that our joint model indeed does better than an equivalent pipeline system. However, usually most joint approaches come at a price: efficiency. Interestingly, in our case we observe the opposite: our joint model is actually faster than the pipeline. This can be seen in table 6, where we list the time it took for several different system to process the WSJ and Brown test corpus, respectively. When we compare the times for the bottom-up model to those of the pipeline, we note that the joint model is twice as fast. While the individual stages within the pipeline may be faster than the joint system (even when we sum up inference times), extracting results from one system and feeding them into another creates overhead which offsets this potential reduction.

Table 6 also lists the run-time of a bottom-up system that solves the inference problem by fully grounding the Markov Network that the Markov Logic (ML) model describes, mapping this network to an Integer Linear Program, and finding the most likely assignment using an ILP solver. This system (Bottom-up (-CPI)) is four times slower than the equivalent system that uses Cutting Plane Inference (Bottom-up). This suggests that if we were to implement the same joint model using ILP instead of ML, our system would either be significantly slower, or we would need to implement a Cutting Plane algorithm for the corresponding ILP formulation—when we use ML this algorithm comes "for free".

System	WSJ	Brown
Full	9.2m	1.5m
Full (-CPI)	38.4m	7.47m
Bottom-up	9.5m	1.6m
Bottom-up (-CPI)	38.8m	6.9m
Pipeline	18.9m	2.9m

Table 6: Testing times for full model and bottom-up when CPI algorithm is not used. The WSJ test set contains 2414 sentences, the Brown test set 426. Our best systems thus takes on average 230ms per WSJ sentence (on a 2.4Ghz system).

7 Conclusion

In this paper we have presented a Markov Logic Network that jointly models all predicate identification, argument identification and classification and sense disambiguation decisions for a sentence. We have shown that this approach is competitive, in particular if we consider that our input parses are significantly worse than those of the top CoNLL 2008 systems.

We demonstrated the benefit of jointly predicting senses and semantic arguments when compared to a pipeline system that first picks arguments and then senses. We also showed that by modelling whether a token is an argument of some predicate and jointly picking arguments for all predicates of a sentence, further improvements can be achieved.

Finally, we demonstrated that our system is efficient, despite following a global approach. This efficiency was also shown to stem from the first order inference method our Markov Logic engine applies.

Acknowledgements

The authors are grateful to Mihai Surdeanu for providing the version of the corpus used in this work.

References

Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of NAACL-2000*, 2000.

Koby Crammer and Yoram Singer. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991, 2003.

Richard Johansson and Pierre Nugues. Dependency-based semantic role labeling of propbank. In *Proceedings of EMNLP-2008.*, 2008.

Lluís Márquez, Xavier Carreras, Ken Litkowski, and Suzanne Stevenson. Semantic role labeling. *Computational Linguistics*, 34(2), 2008. Introduction to the Special Issue on Semantic Role Labeling.

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

V. Punyakanok, D. Roth, and W. Yih. Generalized inference with multiple semantic role labeling systems. In Ido Dagan and Dan Gildea, editors, *CoNLL '05: Proceedings of the Annual Conference on Computational Natural Language Learning*, pages 181–184, 2005.

Matthew Richardson and Pedro Domingos. Markov logic networks. Technical report, University of Washington, 2005.

Sebastian Riedel. Improving the accuracy and efficiency of map inference for markov logic. In *UAI '08: Proceedings of the Annual Conference on Uncertainty in AI*, 2008.

Sebastian Riedel and Ivan Meza-Ruiz. Collective semantic role labelling with markov logic. In *Conference on Computational Natural Language Learning*, 2008.

P. Singla and P. Domingos. Lifted First-Order Belief Propagation. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2008.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*, 2008.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. Joint learning improves semantic role labeling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Morristown, NJ, USA, 2005.

David Vickrey and Daphne Koller. Applying sentence simplification to the conll-2008 shared task. In *Proceedings of CoNLL-2008.*, 2008.

Nianwen Xue and Martha Palmer. Calibrating features for semantic role labeling. In *EMNLP '04: Proceedings of the Annual Conference on Empirical Methods in Natural Language Processing*, 2004.

Hai Zhao and Chunyu Kit. Parsing syntactic and semantic dependencies with two single-stage maximum entropy models. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, Manchester, England, 2008.

Structured Generative Models for Unsupervised Named-Entity Clustering

Micha Elsner, Eugene Charniak and Mark Johnson
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
{melsner,ec,mj}@cs.brown.edu

Abstract

We describe a generative model for clustering named entities which also models named entity internal structure, clustering related words by role. The model is entirely unsupervised; it uses features from the named entity itself and its syntactic context, and coreference information from an unsupervised pronoun resolver. The model scores 86% on the MUC-7 named-entity dataset. To our knowledge, this is the best reported score for a fully unsupervised model, and the best score for a generative model.

1 Introduction

Named entity clustering is a classic task in NLP, and one for which both supervised and semi-supervised systems have excellent performance (Mikheev et al., 1998; Chinchor, 1998). In this paper, we describe a fully unsupervised system (using no "seed rules" or initial heuristics); to our knowledge this is the best such system reported on the MUC-7 dataset. In addition, the model clusters the words which appear in named entities, discovering groups of words with similar roles such as first names and types of organization. Finally, the model defines a notion of consistency between different references to the same entity; this component of the model yields a significant increase in performance.

The main motivation for our system is the recent success of unsupervised generative models for coreference resolution. The model of Haghighi and Klein (2007) incorporated a latent variable for named entity class. They report a named entity score of 61.2 percent, well above the baseline of 46.4, but still far behind existing named-entity systems.

We suspect that better models for named entities could aid in the coreference task. The easiest way to incorporate a better model is simply to run a supervised or semi-supervised system as a preprocess. To perform joint inference, however, requires an unsupervised generative model for named entities. As far as we know, this work is the best such model.

Named entities also pose another problem with the Haghighi and Klein (2007) coreference model; since it models only the heads of NPs, it will fail to resolve some references to named entities: ("Ford Motor Co.", "Ford"), while erroneously merging others: ("Ford Motor Co.", "Lockheed Martin Co."). Ng (2008) showed that better features for matching named entities– exact string match and an "alias detector" looking for acronyms, abbreviations and name variants– improve the model's performance substantially. Yet building an alias detector is nontrivial (Uryupina, 2004). English speakers know that "President Clinton" is the same person as "Bill Clinton" , not "President Bush". But this cannot be implemented by simple substring matching. It requires some concept of the role of each word in the string. Our model attempts to learn this role information by clustering the words within named entities.

2 Related Work

Supervised named entity recognition now performs almost as well as human annotation in English (Chinchor, 1998) and has excellent performance on other languages (Tjong Kim Sang and De Meulder, 2003). For a survey of the state of the art,

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 164–172,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

see Nadeau and Sekine (2007). Of the features we explore here, all but the pronoun information were introduced in supervised work. Supervised approaches such as Black et al. (1998) have used clustering to group together different nominals referring to the same entity in ways similar to the "consistency" approach outlined below in section 3.2.

Semi-supervised approaches have also achieved notable success on the task. Co-training (Riloff and Jones, 1999; Collins and Singer, 1999) begins with a small set of labeling heuristics and gradually adds examples to the training data. Various co-training approaches presented in Collins and Singer (1999) all score about 91% on a dataset of named entities; the inital labels were assigned using 7 hand-written seed rules. However, Collins and Singer (1999) show that a mixture-of-naive-Bayes generative clustering model (which they call an EM model), initialized with the same seed rules, performs much more poorly at 83%.

Much later work (Evans, 2003; Etzioni et al., 2005; Cucerzan, 2007; Pasca, 2004) relies on the use of extremely large corpora which allow very precise, but sparse features. For instance Etzioni et al. (2005) and Pasca (2004) use web queries to count occurrences of "cities such as X" and similar phrases. Although our research makes use of a fairly large amount of data, our method is designed to make better use of relatively common contextual features, rather than searching for high-quality semantic features elsewhere.

Models of the internal structure of names have been used for cross-document coreference (Li et al., 2004; Bhattacharya and Getoor, 2006) and a goal in their own right (Charniak, 2001). Li et al. (2004) take named entity classes as a given, and develops both generative and discriminative models to detect coreference between members of each class. Their generative model designates a particular mention of a name as a "representative" and generates all other mentions from it according to an editing process. Bhattacharya and Getoor (2006) operates only on authors of scientific papers. Their model accounts for a wider variety of name variants than ours, including misspellings and initials. In addition, they confirm our intuition that Gibbs sampling for inference has insufficient mobility; rather than using a heuristic algorithm as we do (see section 3.5), they use a data-driven block sampler. Charniak (2001) uses a Markov chain to generate 6 different components of people's names, again assuming that the class of personal names can be pre-distinguished using a name list. He infers coreference relationships between similar names appearing in the same document, using the same notion of consistency between names as our model. As with our model, the clusters found are relatively good, although with some mistakes even on frequent items (for example, "John" is sometimes treated as a descriptor like "Secretary").

3 System Description

Like Collins and Singer (1999), we assume that the named entities have already been correctly extracted from the text, and our task is merely to label them. We assume that all entities fit into one of the three MUC-7 categories, LOC (locations), ORG (organizations), and PER (people). This is an oversimplification; Collins and Singer (1999) show that about 12% of examples do not fit into these categories. However, while using the MUC-7 data, we have no way to evaluate on such examples.

As a framework for our models, we adopt *adaptor grammars* (Johnson et al., 2007), a framework for non-parametric Bayesian inference over context-free grammars. Although our system does not require the full expressive power of PCFGs, the adaptor grammar framework allows for easy development of structured priors, and supplies a flexible generic inference algorithm. An adaptor grammar is a hierarchical Pitman-Yor process (Pitman and Yor, 1997). The grammar has two parts: a base PCFG and a set of *adapted* nonterminals. Each adapted nonterminal is a Pitman-Yor process which expands either to a previously used subtree or to a sample from the base PCFG. The end result is a posterior distribution over PCFGs and over parse trees for each example in our dataset.

Each of our models is an adaptor grammar based on a particular base PCFG where the top nonterminal of each parse tree represents a named entity class.

3.1 Core NP Model

We begin our analysis by reducing each named-entity reference to the contiguous substring of

$$
\begin{aligned}
ROOT &\rightarrow NE_0 | NE_1 | NE_2 \\
NE_0 &\rightarrow (NE_0^0)(NE_0^1)(NE_0^2)(NE_0^3)(NE_0^4) \\
*NE_0^0 &\rightarrow Words \\
*Words &\rightarrow Word\ (Words) \\
Word &\rightarrow Bill \ldots
\end{aligned}
$$

Figure 1: Part of the grammar for core phrases. (Parentheses) mark optional nonterminals. *Starred nonterminals are adapted.

$$
\begin{aligned}
ROOT &\rightarrow NE_0 | NE_1 | NE_2 \\
NE_0 &\rightarrow E_{00} | E_{01} \ldots E_{0k} \\
E_{00} &\rightarrow (E_{00}^0)(E_{00}^1)(E_{00}^2)(E_{00}^3)(E_{00}^4) \\
**E_{00}^0 &\rightarrow NE_0^0 \\
*NE_0^0 &\rightarrow Words \ldots
\end{aligned}
$$

Figure 2: Part of the consistency-enforcing grammar for core phrases. There are an infinite number of entities E_{xk}, all with their own lexical symbols. Each lexical symbol E_{xk}^i expands to a single NE_x^i.

proper nouns which surrounds its head, which we call the core (Figure 1). To analyze the core, we use a grammar with three main symbols (NE_x), one for each named entity class x. Each class has an associated set of lexical symbols, which occur in a strict order (NE_x^i is the ith symbol for class x). We can think of the NE^i as the semantic parts of a proper name; for people, NE_{PER}^0 might generate titles and NE_{PER}^1 first names. Each NE^i is adapted, and can expand to any string of words; the ability to generate multiple words from a single symbol is useful both because it can learn to group collocations like "New York" and because it allows the system to handle entities longer than four words. However, we set the prior on multi-word expansions very low, to avoid degenerate solutions where most phrases are analyzed with a single symbol. The system learns a separate probability for each ordered subset of the NE^i (for instance the rule $NE_0 \rightarrow NE_0^0\, NE_0^2\, NE_0^4$), so that it can represent constraints on possible references; for instance, a last name can occur on its own, but not a title.

3.2 Consistency Model

This system captures some of our intuitions about core phrases, but not all: our representation for "Bill Clinton" does not share any information with "President Bill Clinton" except the named-entity class. To remedy this, we introduce a set of "entity" nonterminals E_k, which enforce a weak notion of consistency. We follow Charniak (2001) in assuming that two names are consistent (can be references to the same entity) if they do not have different expansions for any lexical symbol. In other words, a particular entity $E_{PER,Clinton}$ has a title $E_{PER,Clinton}^0 =$ "President", a first name $E_{PER,Clinton}^1 =$ "Bill" etc. These are generated from the class-specific distributions, for instance $E_{PER,Clinton}^0 \sim E_{PER}^0$, which we intend to be a distribution over titles in general.

The resulting grammar is shown in Figure 2; the prior parameters for the entity-specific symbols E_{xk}^i are fixed so that, with overwhelming probability, only one expansion occurs. We can represent any fixed number of entities E_k with a standard adaptor grammar, but since we do not know the correct number, we must extend the adaptor model slightly to allow for an unbounded number. We generate the E_k from a Chinese Restaurant process prior. (General grammars with infinite numbers of nonterminals were studied by (Liang et al., 2007b)).

3.3 Modifiers, Prepositions and Pronouns

Next, we introduce two types of context information derived from Collins and Singer (1999): nominal modifiers and prepositional information. A nominal modifier is either the head of an appositive phrase ("Maury Cooper, a vice **president**") or a non-proper prenominal ("**spokesman** John Smith")[1]. If the entity is the complement of a preposition, we extract the preposition and the head of the governing NP ("a federally funded sewage **plant in** Georgia"). These are added to the grammar at the named-entity class level (separated from the core by a special punctuation symbol).

Finally, we add information about pronouns and wh-complementizers (Figure 3). Our pronoun information is derived from an unsupervised coreference algorithm which does not use named entity informa-

[1] We stem modifiers with the Porter stemmer.

$$
\begin{aligned}
ROOT \rightarrow & Modifiers_0 \ \#\ NE_0 \ \# \\
& Prepositions_0 \ \#\ Pronouns_0 \ \# \\
& \ldots \\
Pronouns_0 \rightarrow & Pronoun_0\ Pronouns_0 \\
Pronouns_0 \rightarrow & \\
Pronoun_0 \rightarrow & \mathbf{pers} | loc | org | any \\
pers \rightarrow & i | he | she | who | me \ldots \\
loc \rightarrow & where | which | it | its \\
org \rightarrow & which | it | they | we \ldots
\end{aligned}
$$

Figure 3: A fragment of the full grammar. The symbol # represents punctuation between different feature types. The prior for class 0 is concentrated around **personal** pronouns, although other types are possible.

attack airlift airlift rescu # wing # of-commander of-command with-run # #
air-india # #
abels # # it
gaudreau # # they he
priddy # # he
spokesman bird bird bird director bird ford clinton director bird # johnson # before-hearing to-happened of-cartoon on-pressure under-medicare to-according to-allied with-stuck of-government of-photographs of-daughter of-photo for-embarrassing under-instituted about-allegations for-worked before-hearing to-secretary than-proposition of-typical # he he his he my himself his he he he he i he his his i i i he his #

Figure 4: Some merged examples from an input file. (# separates different feature types.)

tion (Charniak and Elsner, 2009). This algorithm uses EM to learn a generative model with syntactic, number and gender parameters. Like Haghighi and Klein (2007), we give our model information about the basic types of pronouns in English. By setting up the base grammar so that each named-entity class prefers to associate to a single type of pronoun, we can also determine the correspondence between our named-entity symbols and the actual named-entity labels– for the models without pronoun information, this matching is arbitrary and must be inferred during the evaluation process.

3.4 Data Preparation

To prepare data for clustering with our system, we first parse it with the parser of Charniak and Johnson (2005). We then annotate pronouns with Charniak and Elsner (2009). For the evaluation set, we use the named entity data from MUC-7. Here, we extract all strings in <ne> tags and determine their cores, plus any relevant modifiers, governing prepositions and pronouns, by examining the parse trees. In addition, we supply the system with additional data from the North American News Corpus (NANC). Here we extract all NPs headed by proper nouns.

We then process our data by merging all examples with the same core; some merged examples from our dataset are shown in Figure 4. When two examples are merged, we concatenate their lists of modifiers, prepositions and pronouns (capping the length of each list at 20 to keep inference tractable). For instance, "air-india" has no features outside the core, while "wing" has some nominals ("attack" &c.) and some prepositions ("commander-of" &c.). This merging is useful because it allows us to do inference based on types rather than tokens (Goldwater et al., 2006). It is well known that, to interpolate between types and tokens, Hierarchical Dirichlet Processes (including adaptor grammars) require a deeper hierarchy, which slows down inference and reduces the mobility of sampling schemes. By merging examples, we avoid using this more complicated model. Each merged example also represents many examples from the training data, so we can summarize features (such as modifiers) observed throughout a large input corpus while keeping the size of our input file small.

To create an input file, we first add all the MUC-7 examples. We then draw additional examples from NANC, ranking them by how many features they have, until we reach a specified number (larger datasets take longer, but without enough data, results tend to be poor).

3.5 Inference

Our implementation of adaptor grammars is a modified version of the Pitman-Yor adaptor grammar

sampler[2], altered to deal with the infinite number of entities. It carries out inference using a Metropolis-within-Gibbs algorithm (Johnson et al., 2007), in which it repeatedly parses each input line using the CYK algorithm, samples a parse, and proposes this as the new tree.

To do Gibbs sampling for our consistency-enforcing model, we would need to sample a parse for an example from the posterior over every possible entity. However, since there are thousands of entities (the number grows roughly linearly with the number of merged examples in the data file), this is not tractable. Instead, we perform a restricted Gibbs sampling search, where we enumerate the posterior only for entities which share a word in their core with the example in question. In fact, if the shared word is very common (occuring in more than .001 of examples), we compute the posterior for that entity only .05 of the time[3]. These restrictions mean that we do not compute the exact posterior. In particular, the actual model allows entities to contain examples with no words in common, but our search procedure does not explore these solutions.

For our model, inference with the Gibbs algorithm seems to lack mobility, sometimes falling into very poor local minima from which it does not seem to escape. This is because, if there are several references to the same named entity with slightly different core phrases, once they are all assigned to the wrong class, it requires a low-probability series of individual Gibbs moves to pull them out. Similarly, the consistency-enforcing model generally does not fully cluster references to common entities; there are usually several "Bill Clinton" clusters which it would be best to combine, but the sequence of moves that does so is too improbable. The data-merging process described above is one attempt to improve mobility by reducing the number of duplicate examples. In addition, we found that it was a better use of CPU time to run multiple samplers with different initialization than to perform many iterations. In the experiments below, we use 20 chains, initializing with 50 iterations without using consistency, then 50 more using the consistency model, and evaluate the last sample from each. We discard the 10 samples with worst log-likelihood and report the average score for the other 10.

3.6 Parameters

In addition to the base PCFG itself, the system requires a few hyperparameter settings: Dirichlet priors for the rule weights of rules in the base PCFG. Pitman-Yor parameters for the adapted nonterminals are sampled from vague priors using a slice sampler (Neal, 2003). The prior over core words was set to the uniform distribution (Dirichlet 1.0) and the prior for all modifiers, prepositions and pronouns to a sparse value of .01. Beyond setting these parameters to a priori reasonable values, we did not optimize them. To encourage the system to learn that some lexical symbols were more common than others, we set a sparse prior over expansions to symbols[4]. There are two really important hyperparameters: an extremely biased prior on class-to-pronoun-type probabilities (1000 for the desired class, .0001 for everything else), and a prior of .0001 for the $Word \rightarrow Word\ Words$ rule to discourage symbols expanding to multiword strings.

4 Experiments

We performed experiments on the named entity dataset from MUC-7 (Chinchor, 1998), using the training set as development data and the formal test set as test data. The development set has 4936 named entities, of which 1575 (31.9%) are locations, 2096 (42.5%) are organizations and 1265 (25.6%) people. The test set has 4069 named entities, 1321 (32.5%) locations, 1862 (45.8%) organizations and 876 (21.5%) people[5]. We use a baseline which gives all named entities the same label; this label is mapped to "organization".

In most of our experiments, we use an input file of 40000 lines. For dev experiments, the labeled data contributes 1585 merged examples; for test experiments, only 1320. The remaining lines are derived

[2]Available at http://www.cog.brown.edu/ mj/Software.htm

[3]We ignore the corresponding Hastings correction, as in practice it leads to too many rejections.

[4]Expansions that used only the middle three symbols $NE_x^{1,2,3}$ got a prior of .005, expansions whose outermost symbol was $NE_x^{0,4}$ got .0025, and so forth. This is not so important for our final system, which has only 5 symbols, but was designed during development to handle systems with up to 10 symbols.

[5]10 entities are labeled location|organization; since this fraction of the dataset is insignificant we score them as wrong.

Model	Accuracy
Baseline (All Org)	42.5
Core NPs (no consistency)	45.5
Core NPs (consistency)	48.5
Context Features	83.3
Pronouns	87.1

Table 1: Accuracy of various models on development data.

Model	Accuracy
Baseline (All Org)	45.8
Pronouns	86.0

Table 2: Accuracy of the final model on test data.

using the process described in section 3.4 from 5 million words of NANC.

To evaluate our results, we map our three induced labels to their corresponding gold label, then count the overlap; as stated, this mapping is predictably encoded in the prior when we use the pronoun features. Our experimental results are shown in Table 1. All models perform above baseline, and all features contribute significantly to the final result. Test results for our final model are shown in Table 2.

A confusion matrix for our highest-likelihood test solution is shown as Figure 5. The highest confusion class is "organization", which is confused most often with "location" but also with "person". "location" is likewise confused with "organization". "person" is the easiest class to identify– we believe this explains the slight decline in performance from dev to test, since dev has proportionally more people.

Our mapping from grammar symbols to words appears in Table 3; the learned prepositional and modifier information is in Table 4. Overall the results are good, but not perfect; for instance, the $Pers$ states are mostly interpretable as a sequence of title - first name - middle name or initial - last name - last name or post-title (similar to (Charniak, 2001)). The organization symbols tend to put nationalities and other modifiers first, and end with institutional types like "inc." or "center", although there is a similar (but smaller) cluster of types at Org^2, suggesting the system has incorrectly found two analyses for these names. Location symbols seem to put entities with a single, non-analyzable name into Loc^2, and use symbols 0, 1 and 3 for compound names. Loc^4 has been recruited for time expressions, since our NANC dataset includes many of these, but we failed to account for them in the model. Since they appear in a single class here, we are optimistic that they could be clustered separately if another class and some appropriate features were added to the prior. Some errors do appear ("supreme court" and "house" as locations, "minister" and "chairman" as middle names, "newt gingrich" as a multiword phrase). The table also reveals an unforeseen issue with the parser: it tends to analyze the dateline beginning a news story along with the following NP ("WASHINGTON Bill Clinton said..."). Thus common datelines ("washington", "new york" and "los angeles") appear in state 0 for each class.

	loc	*org*	*per*
LOC	1187	97	37
ORG	223	1517	122
PER	36	20	820

Figure 5: Confusion matrix for highest-likelihood test run. Gold labels in CAPS, induced labels *italicized*. Organizations are most frequently confused.

5 Discussion

As stated above, we aim to build an unsupervised generative model for named entity clustering, since such a model could be integrated with unsupervised coreference models like Haghighi and Klein (2007) for joint inference. To our knowledge, the closest existing system to such a model is the EM mixture model used as a baseline in Collins and Singer (1999). Our system improves on this EM system in several ways. While they initialize with minimal supervision in the form of 7 seed heuristics, ours is fully unsupervised. Their results cover only examples which have a prepositional or modifier feature; we adopt these features from their work, but label all entities in the predefined test set, including those that appear without these features. Finally, as discussed, we find the "person" category to be the easiest to label. 33% of the test items in Collins and Singer (1999) were people, as opposed to 21% of ours. However, even without the pronoun features, that is, using the same feature set, our system scores equivalently to the EM model, at 83% (this score is

$Pers^0$	$Pers^1$	$Pers^2$	$Pers^3$	$Pers^4$
rep.	john (767)	minister	brown	jr.
sen. (256)	robert (495)	j.	smith (97)	a
washington	david	john (242)	b	smith (111)
dr.	michael	l.	johnson	iii
los angeles	james	chairman	newt gingrich	williams
senate	president	e.	king	wilson
house	richard	m.	miller	brown
new york	william (317)	william (173)	kennedy	clinton
president	sen. (236)	robert (155)	martin	simpson
republican	george	r.	davis	b
Org^0	Org^1	Org^2	Org^3	Org^4
american (137)	national	university	research	association
washington	american (182)	inc. (166)	medical	center
washington the	new york	corp. (156)	news	inc. (257)
national	international (136)	college	health	corp. (252)
first	public	institute (87)	services	co.
los angeles	united	group	communications	committee
new	house	hospital	development	institute
royal	federal	museum	policy	council
british	home	press	affairs	fund
california	world	international (61)	defense	act
Loc^0	Loc^1	Loc^2	Loc^3	Loc^4
washington (92)	the	texas	county	monday
los angeles	st.	new york	city	thursday
south	new	washington (22)	beach	river (57)
north	national (69)	united states	valley	tuesday
old	east (65)	baltimore	island	wednesday
grand	mount	california	river (71)	hotel
black	fort	capitol	park	friday
west (22)	west (56)	christmas	bay	hall
east (21)	lake	bosnia	house	center
haiti	great	san juan	supreme court	building

Table 3: 10 most common words for each grammar symbol. Words which appear in multiple places have observed counts indicated in parentheses.

Pers-gov	Pers-mod	Org-gov	Org-mod	Loc-gov	Loc-mod
according-to (1044)	director	president-of	$	university-of	calif.
played-by	spokesman	chairman-of	giant	city-of	newspap[er]
directed-by	leader	director-of	opposit[e]	from-to	state
led-by	presid[ent]	according-to (786)	group	town-of	downtown
meeting-with	attorney	professor-at	pp	state-of	n.y.
from-to	candid[ate]	head-of	compan[y]	center-in	warrant
met-with	lawyer	department-of	journal	out-of	va.
letter-to	chairman	member-of	firm	is-in	fla.
secretary-of	counsel	members-of	state	house-of	p.m.
known-as	actor	spokesman-for	agenc[y]	known-as	itself

Table 4: 10 most common prepositional and modifier features for each named entity class. Modifiers were Porter-stemmed; for clarity a reconstructed stem is shown in brackets.

on dev, 25% people). When the pronoun features are added, our system's performance increases to 86%, significantly better than the EM system.

One motivation for our use of a structured model which defined a notion of consistency between entities was that it might allow the construction of an unsupervised alias detector. According to the model, two entities are consistent if they are in the same class, and do not have conflicting assignments of words to lexical symbols. Results here are at best equivocal. The model is reasonable at passing basic tests– "Dr. Seuss" is not consistent with "Dr. Strangelove", "Dr. Quinn" etc, despite their shared title, because the model identifies the second element of each as a last name. Also correctly, "Dr. William F. Gibson" is judged consistent with "Dr. Gibson" and "Gibson" despite the missing elements. But mistakes are commonplace. In the "Gibson" case, the string "William F." is misanalyzed as a multiword string, making the name inconsistent with "William Gibson"; this is probably the result of a search error, which, as we explained, Gibbs sampling is unlikely to correct. In other cases, the system clusters a family group together under a single "entity" nonterminal by forcing their first names into inappropriate states, for instance assigning $Pers^1$ Bruce, $Pers^2$ Ellen, $Pers^3$ Jarvis, where $Pers^2$ (usually a middle name) actually contains the first name of a different individual. To improve this aspect of our system, we might incorporate name-specific features into the prior, such as abbreviations and the concept of a family name. The most critical improvement, however, would be integration with a generative coreference system, since the document context probably provides hints about which entities are and are not coreferent.

The other key issue with our system is inference. Currently we are extremely vulnerable to falling into local minima, since the complex structure of the model can easily lock a small group of examples into a poor configuration. (The "William F. Gibson" case above seems to be one of these.) In addition to the block sampler used by Bhattacharya and Getoor (2006), we are investigating general-purpose split-merge samplers (Jain and Neal, 2000) and the permutation sampler (Liang et al., 2007a). One interesting question is how well these samplers perform when faced with thousands of clusters (entities).

Despite these issues, we clearly show that it is possible to build a good model of named entity class while retaining compatibility with generative systems and without supervision. In addition, we do a reasonable job learning the latent structure of names in each named entity class. Our system improves over the latent named-entity tagging in Haghighi and Klein (2007), from 61% to 87%. This suggests that it should indeed be possible to improve on their coreference results without using a supervised named-entity model. How much improvement is possible in practice, and whether joint inference can also improve named-entity performance, remain interesting questions for future work.

Acknowledgements

We thank three reviewers for their comments, and NSF for support via grants 0544127 and 0631667.

References

Indrajit Bhattacharya and Lise Getoor. 2006. A latent dirichlet model for unsupervised entity resolution. In *The SIAM International Conference on Data Mining (SIAM-SDM)*, Bethesda, MD, USA.

William J. Black, Fabio Rinaldi, and David Mowatt. 1998. Facile: Description of the ne system used for muc-7. In *In Proceedings of the 7th Message Understanding Conference*.

Eugene Charniak and Micha Elsner. 2009. EM works for pronoun anaphora resolution. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*, Athens, Greece.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *Proc. of the 2005 Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 173–180.

Eugene Charniak. 2001. Unsupervised learning of name structure from coreference data. In *NAACL-01*.

Nancy A. Chinchor. 1998. Proceedings of the Seventh Message Understanding Conference (MUC-7) named entity task definition. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, page 21 pages, Fairfax, VA, April. version 3.5, http://www.itl.nist.gov/iaui/894.02/related_projects/muc/.

Michael Collins and Yorav Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of EMNLP 99*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL*, pages 708–716, Prague, Czech Republic, June. Association for Computational Linguistics.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana maria Popescu, Tal Shaked, Stephen Soderl, Daniel S. Weld, and Er Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165:91–134.

Richard Evans. 2003. A framework for named entity recognition in the open domain. In *Proceedings of Recent Advances in Natural Language Processing (RANLP-2003)*, pages 137 – 144, Borovetz, Bulgaria, September.

Sharon Goldwater, Tom Griffiths, and Mark Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems (NIPS) 18*.

Aria Haghighi and Dan Klein. 2007. Unsupervised coreference resolution in a nonparametric Bayesian model. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 848–855. Association for Computational Linguistics.

Sonia Jain and Radford M. Neal. 2000. A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13:158–182.

Mark Johnson, Tom L. Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of NAACL 2007*.

Xin Li, Paul Morie, and Dan Roth. 2004. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *AAAI*, pages 419–424.

Percy Liang, Michael I. Jordan, and Ben Taskar. 2007a. A permutation-augmented sampler for DP mixture models. In *Proceedings of ICML*, pages 545–552, New York, NY, USA. ACM.

Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007b. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of EMNLP-CoNLL*, pages 688–697, Prague, Czech Republic, June. Association for Computational Linguistics.

A. Mikheev, C. Grover, and M. Moens. 1998. Description of the LTG System Used for MUC-7. In *Proceedings of the 7th Message Understanding Conference (MUC-7)*, Fairfax, Virginia.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Journal of Linguisticae Investigationes*, 30(1).

Radford M. Neal. 2003. Slice sampling. *Annals of Statistics*, 31:705–767.

Vincent Ng. 2008. Unsupervised models for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 640–649, Honolulu, Hawaii, October. Association for Computational Linguistics.

Marius Pasca. 2004. Acquisition of categorized named entities for web search. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 137–145, New York, NY, USA. ACM.

Jim Pitman and Marc Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Ann. Probab.*, 25:855–900.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 472–479. AAAI.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Olga Uryupina. 2004. Evaluating name-matching for coreference resolution. In *Proceedings of LREC 04*, Lisbon.

Hierarchical Dirichlet Trees for Information Retrieval

Gholamreza Haffari
School of Computing Sciences
Simon Fraser University
ghaffar1@cs.sfu.ca

Yee Whye Teh
Gatsby Computational Neuroscience
University College London
ywteh@gatsby.ucl.ac.uk

Abstract

We propose a principled probabilisitc framework which uses trees over the vocabulary to capture similarities among terms in an information retrieval setting. This allows the retrieval of documents based not just on occurrences of specific query terms, but also on similarities between terms (an effect similar to query expansion). Additionally our principled generative model exhibits an effect similar to inverse document frequency. We give encouraging experimental evidence of the superiority of the hierarchical Dirichlet tree compared to standard baselines.

1 Introduction

Information retrieval (IR) is the task of retrieving, given a query, the documents relevant to the user from a large quantity of documents (Salton and McGill, 1983). IR has become very important in recent years, with the proliferation of large quantities of documents on the world wide web. Many IR systems are based on some relevance score function $R(j, q)$ which returns the relevance of document j to query q. Examples of such relevance score functions include term frequency-inverse document frequency (tf-idf) and Okapi BM25 (Robertson et al., 1992).

Besides the effect that documents containing more query terms should be more relevant (term frequency), the main effect that many relevance scores try to capture is that of inverse document frequency: the importance of a term is inversely related to the number of documents that it appears in, i.e. the popularity of the term. This is because popular terms, e.g. common and stop words, are often uninformative, while rare terms are often very informative. Another important effect is that related or co-occurring terms are often useful in determining the relevance of documents. Because most relevance scores do not capture this effect, IR systems resort to techniques like query expansion which includes synonyms and other morphological forms of the original query terms in order to improve retrieval results; e.g. (Riezler et al., 2007; Metzler and Croft, 2007).

In this paper we explore a probabilistic model for IR that simultaneously handles both effects in a principled manner. It builds upon the work of (Cowans, 2004) who proposed a hierarchical Dirichlet document model. In this model, each document is modeled using a multinomial distribution (making the bag-of-words assumption) whose parameters are given Dirichlet priors. The common mean of the Dirichlet priors is itself assumed random and given a Dirichlet hyperprior. (Cowans, 2004) showed that the shared mean parameter induces sharing of information across documents in the corpus, and leads to an inverse document frequency effect.

We generalize the model of (Cowans, 2004) by replacing the Dirichlet distributions with Dirichlet tree distributions (Minka, 2003), thus we call our model the *hierarchical Dirichlet tree*. Related terms are placed close by in the vocabulary tree, allowing the model to take this knowledge into account when determining document relevance. This makes it unnecessary to use ad-hoc query expansion methods, as related words such as synonyms will be taken into account by the retrieval rule. The structure of the tree is learned from data in an unsupervised fashion, us-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 173–181,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

ing a variety of agglomerative clustering techniques.

We review the hierarchical Dirichlet document (HDD) model in section 2, and present our proposed hierarchical Dirichlet tree (HDT) document model in section 3. We describe three algorithms for constructing the vocabulary tree in section 4, and give encouraging experimental evidence of the superiority of the hierarchical Dirichlet tree compared to standard baselines in section 5. We conclude the paper in section 6.

2 Hierarchical Dirichlet Document Model

The probabilistic approach to IR assumes that each document in a collection can be modeled probabilistically. Given a query q, it is further assumed that relevant documents j are those with highest generative probability $p(q|j)$ for the query. Thus given q the relevance score is $R(j,q) = p(q|j)$ and the documents with highest relevance are returned.

Assume that each document is a bag of words, with document j modeled as a multinomial distribution over the words in j. Let V be the terms in the vocabulary, n_{jw} be the number of occurrences of term $w \in V$ in document j, and $\theta^{\text{flat}}_{jw}$ be the probability of w occurring in document j (the superscript "flat" denotes a flat Dirichlet as opposed to our proposed Dirichlet tree). (Cowans, 2004) assumes the following hierarchical Bayesian model for the document collection:

$$\begin{aligned} \boldsymbol{\theta}^{\text{flat}}_0 &= (\theta^{\text{flat}}_{0w})_{w\in V} \sim \text{Dirichlet}(\gamma \mathbf{u}) \\ \boldsymbol{\theta}^{\text{flat}}_j &= (\theta^{\text{flat}}_{jw})_{w\in V} \sim \text{Dirichlet}(\alpha \boldsymbol{\theta}^{\text{flat}}_0) \\ \mathbf{n}_j &= (n_{jw})_{w\in V} \sim \text{Multinomial}(\boldsymbol{\theta}^{\text{flat}}_j) \end{aligned} \tag{1}$$

In the above, bold face $\mathbf{a} = (a_w)_{w\in V}$ means that $\mathbf{a}$ is a vector with $|V|$ entries indexed by $w \in V$, and $\mathbf{u}$ is a uniform distribution over V. The generative process is as follows (Figure 1(a)). First a vector $\boldsymbol{\theta}^{\text{flat}}_0$ is drawn from a symmetric Dirichlet distribution with concentration parameter γ. Then we draw the parameters $\boldsymbol{\theta}^{\text{flat}}_j$ for each document j from a common Dirichlet distribution with mean $\boldsymbol{\theta}^{\text{flat}}_0$ and concentration parameter α. Finally, the term frequencies of the document are drawn from a multinomial distribution with parameters $\boldsymbol{\theta}^{\text{flat}}_j$.

The insight of (Cowans, 2004) is that because the common mean parameter $\boldsymbol{\theta}^{\text{flat}}_0$ is random, it induces dependencies across the document models in the collection, and this in turn is the mechanism for information sharing among documents. (Cowans, 2004) proposed a good estimate of $\boldsymbol{\theta}^{\text{flat}}_0$:

$$\theta^{\text{flat}}_{0w} = \frac{\gamma/|V| + n_{0w}}{\gamma + \sum_{w\in V} n_{0w}} \tag{2}$$

where n_{0w} is simply the number of documents containing term w, i.e. the document frequency. Integrating out the document parameters $\boldsymbol{\theta}^{\text{flat}}_j$, we see that the probability of query q being generated from document j is:

$$\begin{aligned} p(q|j) &= \prod_{x\in q} \frac{\alpha\theta^{\text{flat}}_{0x} + n_{jx}}{\alpha + \sum_{w\in V} n_{jw}} \\ &= \text{Const} \cdot \prod_{x\in q} \frac{\text{Const} + \frac{n_{jx}}{\gamma/|V| + n_{0x}}}{\alpha + \sum_{w\in V} n_{jw}} \end{aligned} \tag{3}$$

Where Const are terms not depending on j. We see that n_{jx} is term frequency, its denominator $\gamma/|V| + n_{0x}$ is an inverse document frequency factor, and $\alpha + \sum_{w\in V} n_{jw}$ normalizes for document length. The inverse document frequency factor is directly related to the shared mean parameter, in that popular terms x will have high $\theta^{\text{flat}}_{0x}$ value, causing all documents to assign higher probability to x, and down weighting the term frequency. This effect will be inherited by our model in the next section.

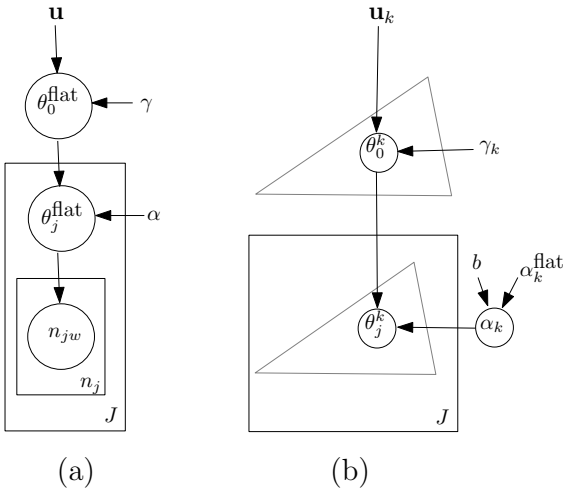


Figure 1: (a) The graphical model representation of the hierarchical Dirichlet document model. (b) The global tree and local trees in hierarchical Dirichlet tree document model. Triangles stand for trees with the same structure, but different parameters at each node. The generation of words in each document is not shown.

3 Hierarchical Dirichlet Trees

Apart from the constraint that the parameters should sum to one, the Dirichlet priors in the HDD model do not impose any dependency among the parameters of the resulting multinomial. In other words, the document models cannot capture the notion that related terms tend to co-occur together. For example, this model cannot incorporate the knowledge that if the word 'computer' is seen in a document, it is likely to observe the word 'software' in the same document. We relax the independence assumption of the Dirichlet distribution by using Dirichlet tree distributions (Minka, 2003), which can capture some dependencies among the resulting parameters. This allows relationships among terms to be modeled, and we will see that it improves retrieval performance.

3.1 Model

Let us assume that we have a tree over the vocabulary whose leaves correspond to vocabulary terms. Each internal node k of the tree has a multinomial distribution over its children $C(k)$. Words are drawn by starting at the root of the tree, recursively picking a child $l \in C(k)$ whenever we are in an internal node k, until we reach a leaf of the tree which corresponds to a vocabulary term (see Figure 2(b)). The Dirichlet tree distribution is the product of Dirichlet distributions placed over the child probabilities of each internal node, and serves as a (dependent) prior over the parameters of multinomial distributions over the vocabulary (the leaves).

Our model generalizes the HDD model by replacing the Dirichlet distributions in (1) by Dirichlet tree distributions. At each internal node k, define a hierarchical Dirichlet prior over the choice of the children:

$$\boldsymbol{\theta}_{0k} = (\theta_{0l})_{l \in C(k)} \sim \text{Dirichlet}(\gamma_k \mathbf{u}_k) \qquad (4)$$
$$\boldsymbol{\theta}_{jk} = (\theta_{jl})_{l \in C(k)} \sim \text{Dirichlet}(\alpha_k \boldsymbol{\theta}_{0k})$$

where $\mathbf{u}_k$ is a uniform distribution over the children of node k, and each internal node has its own hyperparameters γ_k and α_k. θ_{jl} is the probability of choosing child l if we are at internal node k. If the tree is degenerate with just one internal node (the root) and all leaves are direct children of the root we recover the "flat" HDD model in the previous section. We call our model the *hierarchical Dirichlet tree* (HDT).

3.2 Inference and Learning

Given a term, the path from the root to the corresponding leaf is unique. Thus given the term frequencies $\mathbf{n}_j$ of document j as defined in (1), the number of times n_{jl} child $l \in C(k)$ was picked at node k is known and fixed. The probability of all words in document j, given the parameters, is then a product of multinomials probabilities over internal nodes k:

$$p(\mathbf{n}_j | \{\boldsymbol{\theta}_{jk}\}) = \prod_k \frac{n_{jk}!}{\prod_{l \in C(k)} n_{jl}!} \prod_{l \in C(k)} \theta_{jl}^{n_{jl}} \qquad (5)$$

The probability of the documents, integrating out the $\boldsymbol{\theta}_{jk}$'s, is:

$$p(\{\mathbf{n}_j\} | \{\boldsymbol{\theta}_{0k}\}) = \qquad (6)$$
$$\prod_j \prod_k \frac{n_{jk}!}{\prod_{l \in C(k)} n_{jl}!} \frac{\Gamma(\alpha_k)}{\Gamma(\alpha_k + n_{jk})} \prod_{l \in C(k)} \frac{\Gamma(\alpha_k \theta_{0l} + n_{jl})}{\Gamma(\alpha_k \theta_{0l})}$$

The probability of a query q under document j, i.e. the relevance score, follows from (3):

$$p(q|j) = \prod_{x \in q} \prod_{(kl)} \frac{\alpha_k \theta_{0l} + n_{jl}}{\alpha_k + n_{jk}} \qquad (7)$$

where the second product is over pairs (kl) where k is a parent of l on the path from the root to x.

The hierarchical Dirichlet tree model we proposed has a large number of parameters and hyperparameters (even after integrating out the $\boldsymbol{\theta}_{jk}$'s), since the vocabulary trees we will consider later typically have large numbers of internal nodes. This over flexibility might lead to overfitting or to parameter regimes that do not aid in the actual task of IR. To avoid both issues, we constrain the hierarchical Dirichlet tree to be centered over the flat hierarchical Dirichlet document model, and allow it to learn only the α_k hyperparameters, integrating out the $\boldsymbol{\theta}_{jk}$ parameters.

We set $\{\boldsymbol{\theta}_{0k}\}$, the hyperparameters of the global tree, so that it induces the same distribution over vocabulary terms as $\boldsymbol{\theta}_0^{\text{flat}}$:

$$\theta_{0l} = \theta_{0l}^{\text{flat}} \qquad \theta_{0k} = \sum_{l \in C(k)} \theta_{0l} \qquad (8)$$

The hyperparameters of the local trees α_k's are estimated using maximum a posteriori learning with likelihood given by (6), and a gamma prior with informative parameters. The density function of a $\mathrm{Gamma}(a, b)$ distribution is

$$g(x; a, b) = \frac{x^{a-1} b^a e^{-bx}}{\Gamma(a)}$$

where the mode happens at $x = \frac{a-1}{b}$. We set the mode of the prior such that the hierarchical Dirichlet tree reduces to the hierarchical Dirichlet document model at these values:

$$\alpha_l^{\text{flat}} = \alpha\theta_{0l}^{\text{flat}} \qquad \alpha_k^{\text{flat}} = \sum_{l \in C(k)} \alpha_l^{\text{flat}} \tag{9}$$

$$\alpha_k \sim \mathrm{Gamma}(b\alpha_k^{\text{flat}} + 1, b)$$

and $b > 0$ is an inverse scale hyperparameter to be tuned, with large values giving a sharp peak around α_k^{flat}. We tried a few values[1] of b and have found that the results we report in the next section are not sensitive to b. This prior is constructed such that if there is insufficient information in (6) the MAP value will simply default back to the hierarchical Dirichlet document model.

We used LBFGS[2] which is a gradient based optimization method to find the MAP values, where the gradient of the likelihood part of the objective function (6) is:

$$\frac{\partial \log p(\{\mathbf{n}_j\}|\{\boldsymbol{\theta}_{0j}\})}{\partial \alpha_k} = \sum_j \Psi(\alpha_k) - \Psi(\alpha_k + n_{jk}) + \sum_{l \in C(k)} \theta_{0l} \Big(\Psi(\alpha_k \theta_{0l} + n_{jl}) - \Psi(\alpha_k \theta_{0l}) \Big)$$

where $\Psi(x) := \partial \log \Gamma(x) / \partial x$ is the digamma function. Because each α_k can be optimized separately, the optimization is very fast (approximately 15-30 minutes in the experiments to follow on a Linux machine with 1.8 GH CPU speed).

4 Vocabulary Tree Structure Learning

The structure of the vocabulary tree plays an important role in the quality of the HDT document model, since it encapsulates the similarities among words captured by the model. In this paper we explored using trees learned in an unsupervised fashion from the training corpus.

Algorithm 1 Greedy Agglomerative Clustering

1: Place m words into m singleton clusters
2: **repeat**
3: Merge the two clusters with highest similarity, resulting in one less cluster
4: If there still are unincluded words, pick one and place it in a singleton cluster, resulting in one more cluster
5: **until** all words have been included and there is only one cluster left

The three methods are all agglomerative clustering algorithms (Duda et al., 2000) with different similarity functions. Initially each vocabulary word is placed in its own cluster; each iteration of the algorithm finds the pair of clusters with highest similarity and merges them, continuing until only one cluster is left. The sequence of merges determines a binary tree with vocabulary words as its leaves.

Using a heap data structure, this basic agglomerative clustering algorithm requires $\mathcal{O}(n^2 \log(n) + sn^2)$ computations where n is the size of the vocabulary and s is the amount of computation needed to compute the similarity between two clusters. Typically the vocabulary size n is large; to speed up the algorithm, we use a greedy version described in Algorithm 1 which restricts the number of cluster candidates to at most $m \ll n$. This greedy version is faster with complexity $\mathcal{O}(nm(\log m + s))$. In the experiments we used $m = 500$.

Distributional clustering (Dcluster) (Pereira et al., 1993) measures similarity among words in terms of the similarity among their local contexts. Each word is represented by the frequencies of various words in a window around each occurrence of the word. The similarity between two words is computed to be a symmetrized KL divergence between the distributions over neighboring words associated with the two words. For a cluster of words the neighboring words are the union of those associated with each word in the cluster. Dcluster has been used extensively in text classification (Baker and McCallum, 1998).

Probabilistic hierarchical clustering (Pcluster)

[1] Of the form 10^i for $i \in \{-2, -1, 0, 1\}$.

[2] We used a C++ re-implementation of Jorge Nocedal's LBFGS library (Nocedal, 1980) from the ALGLIB website: http://www.alglib.net.

(Friedman, 2003). Dcluster associates each word with its local context, as a result it captures both semantic and syntactic relationships among words. Pcluster captures more relevant semantic relationships by instead associating each word with the documents in which it appears. Specifically, each word is associated with a binary vector indexed by documents in the corpus, where a 1 means the word appears in the corresponding document. Pcluster models a cluster of words probabilistically, with the binary vectors being iid draws from a product of Bernoulli distributions. The similarity of two clusters c_1 and c_2 of words is $P(c_1 \cup c_2)/P(c_1)P(c_2)$, i.e. two clusters of words are similar if their union can be effectively modeled using one cluster, relative to modeling each separately. Conjugate beta priors are placed over the parameters of the Bernoulli distributions and integrated out so that the similarity scores are comparable.

Brown's algorithm (Bcluster) (Brown et al., 1990) was originally proposed to build class-based language models. In the 2-gram case, words are clustered such that the class of the previous word is most predictive of the class of the current word. Thus the similarity between two clusters of words is defined to be the resulting mutual information between adjacent classes corrresponding to a sequence of words.

4.1 Operations to Simplify Trees

Trees constructed using the agglomerative hierarchical clustering algorithms described in this section suffer from a few drawbacks. Firstly, because they are binary trees they have large numbers of internal nodes. Secondly, many internal nodes are simply not informative in that the two clusters of words below a node are indistinguishable. Thirdly, Pcluster and Dcluster tend to produce long chain-like branches which significantly slows down the computation of the relevance score.

To address these issues, we considered operations to simplify trees by contracting internal edges of the tree while preserving as much of the word relationship information as possible. Let L be the set of tree leaves and $\tau(a)$ be the distance from node or edge a to the leaves:

$$\tau(a) := \min_{l \in L} \#\{\text{edges between } a \text{ and } l\} \qquad (10)$$

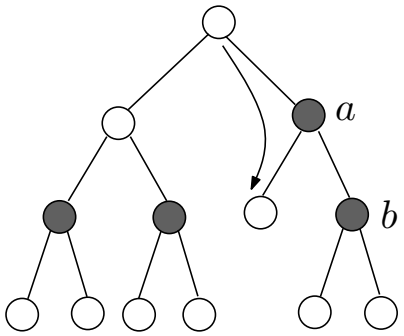


Figure 2: $\tau(\text{root}) = 2$, while $\tau(v) = 1$ for shaded vertices v. Contracting a and b results in both child of b being direct children of a while b is removed.

In the experiments we considered either contracting edges[3] close to the leaves $\tau(a) = 1$ (thus removing many of the long branches described above), or edges further up the tree $\tau(a) \geq 2$ (preserving the informative subtrees closer to the leaves while removing many internal nodes). See Figure 2.

(Miller et al., 2004) cut the BCluster tree at a certain depth k to simplify the tree, meaning every leaf descending from a particular internal node at level k is made an immediate child of that node. They use the tree to get extra features for a discriminative model to tackle the problem of sparsity—the features obtained from the new tree do not suffer from sparsity since each node has several words as its leaves. This technique did not work well for our application so we will not report results using it in our experiments.

5 Experiments

In this section we present experimental results on two IR datasets: Cranfield and Medline[4]. The Cranfield dataset consists of 1,400 documents and 225 queries; its vocabulary size after stemming and removing stop words is 4,227. The Medline dataset contains 1,033 documents and 30 queries with the vocabulary size of 8,800 after stemming and removing stop words. We compare HDT with the flat HDD model and Okapi BM25 (Robertson et al., 1992). Since one of our motivations has been to

[3]Contracting an edge means removing the edge and the adjacent child node and connecting the grandchildren to the parent.

[4]Both datasets can be downloaded from http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections.

Tree	Depth Statistics				Performance			
	Cranfield		Medline		Cranfield		Medline	
	avg / max	total	avg / max	total	avg-pr	top10-pr	avg-pr	top10-pr
BCluster	16.7 / 24	4226	16.4 / 22	8799	0.2675	0.3218	**0.2131**	0.6433
BC contract $\tau \geq 2$	6.2 / 16	3711	5.3 / 14	7473	**0.2685**	0.3147	0.2079	0.6533
BC contract $\tau = 1$	16.1 / 23	3702	15.8 / 22	7672	**0.2685**	0.3204	0.1975	0.6400
DCluster	41.2 / 194	4226	38.1 / 176	8799	0.2552	0.3120	0.1906	0.6300
DC contract $\tau \geq 2$	2.3 / 8	2469	3.3 / 9	5091	0.2555	0.3156	0.1906	0.6167
DC contract $\tau = 1$	40.9 / 194	3648	38.1 / 176	8799	0.2597	0.3129	0.1848	0.6300
PCluster	50.2 / 345	4226	37.1 / 561	8799	0.2613	0.3231	0.1681	0.6633
PC contract $\tau \geq 2$	35.2 / 318	3741	20.4 / 514	7280	0.2624	0.3213	0.1792	0.6767
PC contract $\tau = 1$	33.6 / 345	2246	34.1 / 561	4209	0.2588	**0.3240**	0.1880	0.6633
flat model	1 / 1	1	1 / 1	1	0.2506	0.3089	0.1381	0.6133
BM25	–	–	–	–	0.2566	0.3124	0.1804	0.6567
BM25QueryExp	–	–	–	–	0.2097	0.3191	0.2121	**0.7366**

Table 1: Average precision and Top-10 precision scores of HDT with different trees versus flat model and BM25. The statistics for each tree shows its average/maximum depth of its leaf nodes as well as the number of its total internal nodes. The **bold** numbers highlight the best results in the corresponding columns.

get away from query expansion, we also compare against Okapi BM25 with query expansion. The new terms to expand each query are chosen based on Robertson-Sparck Jones weights (Robertson and Sparck Jones, 1976) from the pseudo relevant documents. The comparison criteria are (i) top-10 precision, and (ii) average precision.

5.1 HDT vs Baselines

All the hierarchical clustering algorithms mentioned in section 4 are used to generate trees, each of which is further post-processed by tree simplification operators described in section 4.1. We consider (i) contracting nodes at higher levels of the hierarchy ($\tau \geq 2$), and (ii) contracting nodes right above the leaves ($\tau = 1$).

The statistics of the trees before and after post-processing are shown in Table 1. Roughly, the Dcluster and BCluster trees do not have long chains with leaves hanging directly off them, which is why their average depths are reduced significantly by the $\tau \geq 2$ simplification, but not by the $\tau = 1$ simplification. The converse is true for Pcluster: the trees have many chains with leaves hanging directly off them, which is why average depth is not reduced as much as the previous trees based on the $\tau \geq 2$ simplification. However the average depth is still reduced significantly compared to the original trees.

Table 1 presents the performance of HDT with different trees against the baselines in terms of the top-10 and average precision (we have bold faced the performance values which are the maximum of each column). HDT with every tree outperforms significantly the flat model in both datasets. More specifically, HDT with (original) BCluster and PCluster trees significantly outperforms the three baselines in terms of both performance measure for the Cranfield. Similar trends are observed on the Medline except here the baseline Okapi BM25 with query expansion is pretty strong[5], which is still outperformed by HDT with BCluster tree.

To further highlight the differences among the methods, we have shown the precision at particular recall points on Medline dataset in Figure 4 for HDT with PCluster tree vs the baselines. As the recall increases, the precision of the PCluster tree significantly outperforms the flat model and BM25. We attribute this to the ability of PCluster tree to give high scores to documents which have words relevant to a query word (an effect similar to query expansion).

5.2 Analysis

It is interesting to contrast the learned α_k's for each of the clustering methods. These α_k's impose cor-

[5]Note that we tuned the parameters of the baselines BM25 with/without query expansion with respect to their performance on the actual retrieval task, which in a sense makes them appear better than they should.

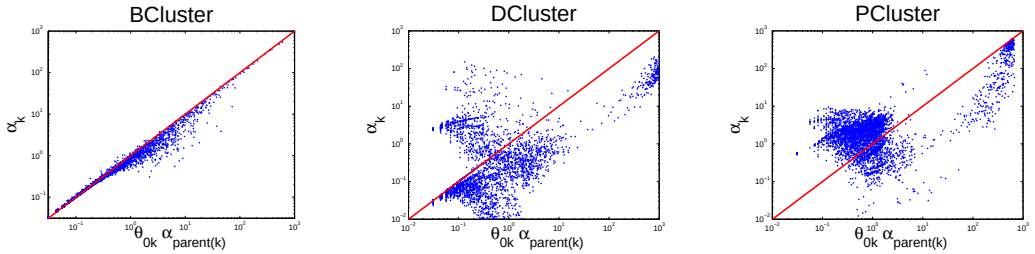


Figure 3: The plots showing the contribution of internal nodes in trees constructed by the three clustering algorithms for the Cranfield dataset. In each plot, a point represent an internal node showing a positive exponent in the node's contribution (i.e. positive correlation among its children) if the point is below $x = y$ line. From left to the right plots, the fraction of nodes below the line is 0.9044, 0.7977, and 0.3344 for a total of 4,226 internal nodes.

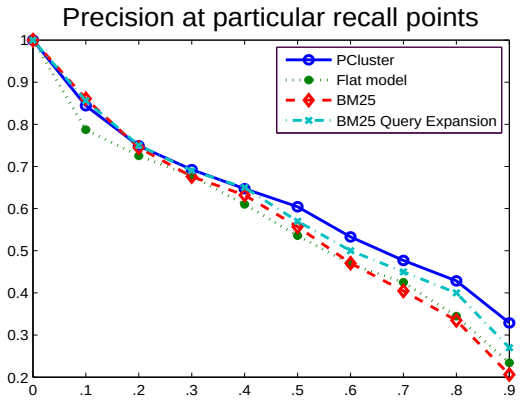


Figure 4: The precision of all methods at particular recall points for the Medline dataset.

relations on the probabilities of the children under k in an interesting fashion. In particular, if we compare α_k to $\theta_{0k}\alpha_{\text{parent}(k)}$, then a larger value of α_k implies that the probabilities of picking one of the children of k (from among all nodes) are positively correlated, while a smaller value of α_k implies negative correlation. Roughly speaking, this is because drawn values of θ_{jl} for $l \in C(k)$ are more likely to be closer to uniform (relative to the flat Dirichlet) thus if we had picked one child of k we will likely pick another child of k.

Figure 3 shows scatter plots of α_k values versus $\theta_{0k}\alpha_{\text{parent}(k)}$ for the internal nodes of the trees. Firstly, smaller values for both tend to be associated with lower levels of the trees, while large values are with higher levels of the trees. Thus we see that PCluster tend to have subtrees of vocabulary terms that are positively correlated with each other—i.e. they tend to co-occur in the same documents. The converse is true of DCluster and BCluster because they tend to put words with the same meaning together, thus to express a particular concept it is enough to select one of the words and not to choose the rest. Figure 5 show some fragments of the actual trees including the words they placed together and α_k parameters learned by HDT model for their internal nodes. Moreover, visual inspection of the trees shows that DCluster can easily misplace words in the tree, which explains its lower performance compared to the other tree construction methods.

Secondly, we observed that for higher nodes of the tree (corresponding generally to larger values of α_k and $\theta_{0k}\alpha_{\text{parent}(k)}$) PCluster α_k's are smaller, thus higher levels of the tree exhibit negative correlation. This is reasonable, since if the subtrees capture positively correlated words, then higher up the tree the different subtrees correspond to clusters of words that do not co-occur together, i.e. negatively correlated.

6 Conclusion and Future Work

We presented a hierarchical Dirichlet tree model for information retrieval which can inject (semantical or syntactical) word relationships as the domain knowledge into a probabilistic model for information retrieval. Using trees to capture word relationships, the model is highly efficient while making use of both prior information about words and their occurrence statistics in the corpus. Furthermore, we investigated the effect of different tree construction algorithms on the model performance.

On the Cranfield dataset, HDT achieves 26.85% for average-precision and 32.40% for top-10 preci-

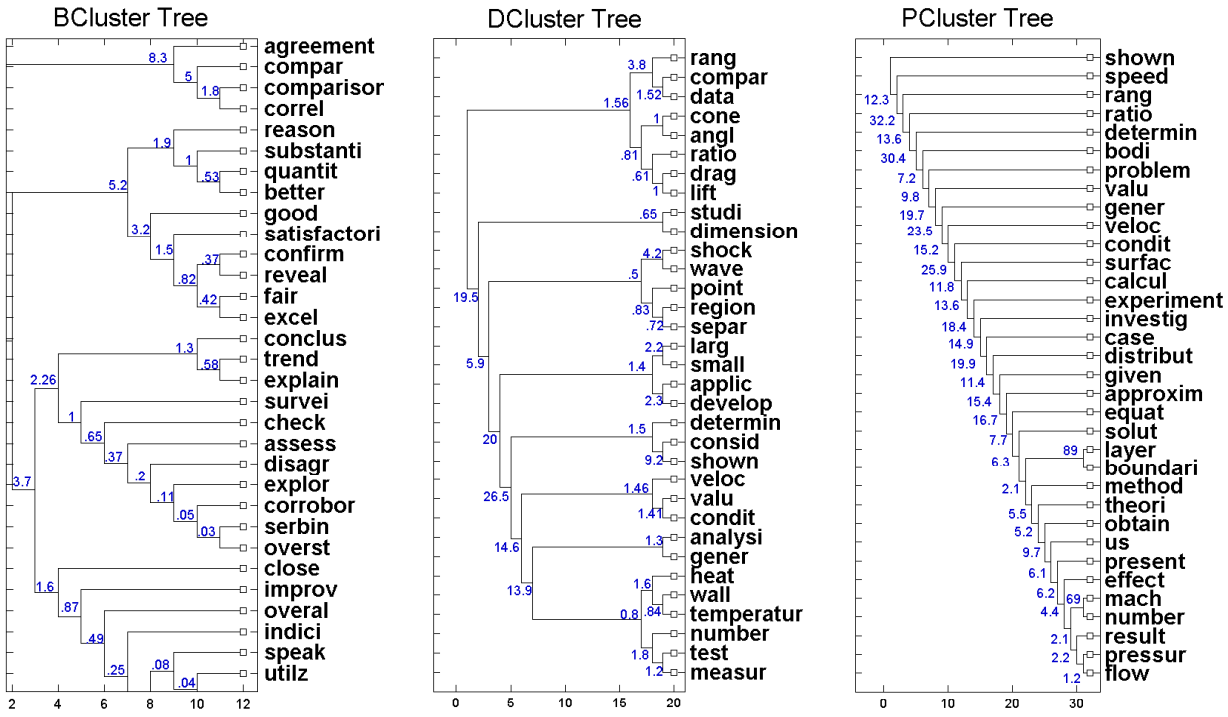


Figure 5: Small parts of the trees learned by clustering algorithms for the Cranfield dataset where the learned α_k for each internal node is written close to it.

sion, and outperforms all baselines including BM25 which gets 25.66% and 31.24% for these two measures. On the Medline dataset, HDT is competitive with BM25 with Query Expansion and outperforms all other baselines. These encouraging results show the benefits of HDT as a principled probabilistic model for information retrieval.

An interesting avenue of research is to construct the vocabulary tree based on WordNet, as a way to inject independent prior knowledge into the model. However WordNet has a low coverage problem, i.e. there are some words in the data which do not exist in it. One solution to this low coverage problem is to combine trees generated by the clustering algorithms mentioned in this paper and WordNet, which we leave as a future work.

References

L. Douglas Baker and Andrew Kachites McCallum. 1998. Distributional clustering of words for text classification. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103.

P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. 1990. Class-based n-gram models of natural language. *Computational Linguistics*.

P. J. Cowans. 2004. Information retrieval using hierarchical dirichlet processes. In *Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval (SIGIR)*.

R. O. Duda, P. E. Hart, and D. G. Stork. 2000. *Pattern Classification*. Wiley-Interscience Publication.

N. Friedman. 2003. Pcluster: Probabilistic agglomerative clustering of gene expression profiles. Available from http://citeseer.ist.psu.edu/668029.html.

Donald Metzler and W. Bruce Croft. 2007. Latent concept expansion using markov random fields. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*.

S. Miller, J. Guinness, and A. Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proceedings of North American Chapter of the Association for Computational Linguistics - Human Language Technologies conference (NAACL HLT)*.

T. Minka. 2003. The dirichlet-tree distribution. Available from http://research.microsoft.com/minka/papers/dirichlet/minka-dirtree.pdf.

J. Nocedal. 1980. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

S. E. Robertson and K. Sparck Jones. 1976. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146.

S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. 1992. Okapi at trec. In *Text REtrieval Conference*, pages 21–30.

G. Salton and M.J. McGill. 1983. *An Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

Phrase-Based Query Degradation Modeling for Vocabulary-Independent Ranked Utterance Retrieval

J. Scott Olsson
HLT Center of Excellence
Johns Hopkins University
Baltimore, MD 21211, USA
solsson@jhu.edu

Douglas W. Oard
College of Information Studies
University of Maryland
College Park, MD 15213, USA
oard@umd.edu

Abstract

This paper introduces a new approach to ranking speech utterances by a system's confidence that they contain a spoken word. Multiple alternate pronunciations, or degradations, of a query word's phoneme sequence are hypothesized and incorporated into the ranking function. We consider two methods for hypothesizing these degradations, the best of which is constructed using factored phrase-based statistical machine translation. We show that this approach is able to significantly improve upon a state-of-the-art baseline technique in an evaluation on held-out speech. We evaluate our systems using three different methods for indexing the speech utterances (using phoneme, phoneme multigram, and word recognition), and find that degradation modeling shows particular promise for locating out-of-vocabulary words when the underlying indexing system is constructed with standard word-based speech recognition.

1 Introduction

Our goal is to find short speech utterances which contain a query word. We accomplish this goal by ranking the set of utterances by our confidence that they contain the query word, a task known as Ranked Utterance Retrieval (RUR). In particular, we are interested in the case when the user's query word can not be anticipated by a Large Vocabulary Continuous Speech Recognizer's (LVCSR) decoding dictionary, so that the word is said to be Out-Of-Vocabulary (OOV).

Rare words tend to be the most informative, but are also most likely to be OOV. When words are OOV, we must use vocabulary-independent techniques to locate them. One popular approach is to search for the words in output from a phoneme recognizer (Ng and Zue, 2000), although this suffers from the low accuracy typical of phoneme recognition. We consider two methods for handling this inaccuracy. First, we compare an RUR indexing system using phonemes with two systems using longer recognition units: words or phoneme multigrams. Second, we consider several methods for handling the recognition inaccuracy in the utterance ranking function itself. Our baseline generative model handles errorful recognition by estimating term frequencies from smoothed language models trained on phoneme lattices. Our new approach, which we call query degradation, hypothesizes many alternate "pronunciations" for the query word and incorporates them into the ranking function. These degradations are *translations* of the lexical phoneme sequence into the errorful recognition language, which we hypothesize using a factored phrase-based statistical machine translation system.

Our speech collection is a set of oral history interviews from the MALACH collection (Byrne et al., 2004), which has previously been used for *ad hoc* speech retrieval evaluations using one-best word level transcripts (Pecina et al., 2007; Olsson, 2008a) and for vocabulary-independent RUR (Olsson, 2008b). The interviews were conducted with survivors and witnesses of the Holocaust, who discuss their experiences before, during, and after the Second World War. Their speech is predominately spontaneous and conversational. It is often also emotional and heavily accented. Because the speech contains many words unlikely to occur within a general purpose speech recognition lexicon, it repre-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 182–190,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

sents an excellent collection for RUR evaluation.

We were graciously permitted to use BBN Technology's speech recognition system *Byblos* (Prasad et al., 2005; Matsoukas et al., 2005) for our speech recognition experiments. We train on approximately 200 hours of transcribed audio excerpted from about 800 unique speakers in the MALACH collection. To provide a realistic set of OOV query words, we use an LVCSR dictionary previously constructed for a different topic domain (broadcast news and conversational telephone speech) and discard all utterances in our acoustic training data which are not covered by this dictionary. New acoustic and language models are trained for each of the phoneme, multigram and word recognition systems.

The output of LVCSR is a *lattice* of recognition hypotheses for each test speech utterance. A lattice is a directed acyclic graph that is used to compactly represent the search space for a speech recognition system. Each node represents a point in time and arcs between nodes indicates a word occurs between the connected nodes' times. Arcs are weighted by the probability of the word occurring, so that the so-called "one-best" path through the lattice (what a system might return as a transcription) is the path through the lattice having highest probability under the acoustic and language models. Each RUR model we consider is constructed using the expected counts of a query word's phoneme sequences in these recognition lattices. We consider three approaches to producing these phoneme lattices, using standard word-based LVCSR, phoneme recognition, and LVCSR using phoneme multigrams. Our word system's dictionary contains about $50{,}000$ entries, while the phoneme system contains 39 phonemes from the ARPABET set.

Originally proposed by Deligne and Bimbot (1997) to model variable length regularities in streams of symbols (e.g., words, graphemes, or phonemes), phoneme multigrams are short sequences of one or more phonemes. We produce a set of "phoneme transcripts" by replacing transcript words with their lexical pronunciation. The set of multigrams is learned by then choosing a maximum-likelihood segmentation of these training phoneme transcripts, where the segmentation is viewed as hidden data in an Expectation-Maximization algorithm. The set of all continuous phonemes occurring between segment boundaries is then chosen as our multigram dictionary. This multigram recognition dictionary contains $16{,}409$ entries.

After we have obtained each recognition lattice, our indexing approach follows that of Olsson (2008b). Namely, for the word and multigram systems, we first expand lattice arcs containing multiple phones to produce a lattice having only single phonemes on its arcs. Then, we compute the expected count of all phoneme n-grams $n \leq 5$ in the lattice. These n-grams and their counts are inserted in our inverted index for retrieval.

This paper is organized as follows. In Section 2 we introduce our baseline RUR methods. In Section 3 we introduce our query degradation approach. We introduce our experimental validation in Section 4 and our results in Section 5. We find that using phrase-based query degradations can significantly improve upon a strong RUR baseline. Finally, in Section 6 we conclude and outline several directions for future work.

2 Generative Baseline

Each method we present in this paper ranks the utterances by the term's estimated frequency within the corresponding phoneme lattice. This general approach has previously been considered (Yu and Seide, 2005; Saraclar and Sproat, 2004), on the basis that it provides a minimum Bayes-risk ranking criterion (Yu et al., Sept 2005; Robertson, 1977) for the utterances. What differs for each method is the particular estimator of term frequency which is used. We first outline our baseline approach, a generative model for term frequency estimation.

Recall that our vocabulary-independent indices contain the expected counts of phoneme sequences from our recognition lattices. Yu and Seide (2005) used these expected phoneme sequence counts to estimate term frequency in the following way. For a query term Q and lattice $\mathcal{L}$, term frequency $\widehat{tf}_G$ is estimated as $\widehat{tf}_G(Q, \mathcal{L}) = P(Q|\mathcal{L}) \cdot N_{\mathcal{L}}$, where $N_{\mathcal{L}}$ is an estimate for the number of words in the utterance. The conditional $P(Q|\mathcal{L})$ is modeled as an order M phoneme level language model,

$$\hat{P}(Q|\mathcal{L}) = \prod_{i=1}^{l} \tilde{P}(q_i|q_{i-M+1}, \ldots, q_{i-1}, \mathcal{L}), \quad (1)$$

so that $\widehat{tf}_G(Q, \mathcal{L}) \approx \hat{P}(Q|\mathcal{L}) \cdot N_{\mathcal{L}}$. The probability of a query phoneme q_j being generated, given that the phoneme sequence $q_{j-M+1}, \ldots, q_{j-1} = q_{j-M+1}^{j-1}$ was observed, is estimated as

$$\tilde{P}(q_j|q_{j-M+1}^{j-1}, \mathcal{L}) = \frac{E_{P_{\mathcal{L}}}[C(q_{j-M+1}^{j})]}{E_{P_{\mathcal{L}}}[C(q_{j-M+1}^{j-1})]}.$$

Here, $E_{P_{\mathcal{L}}}[C(q_{j-M+1}^{j-1})]$ denotes the expected count in lattice $\mathcal{L}$ of the phoneme sequence q_{j-M+1}^{j-1}. We compute these counts using a variant of the forward-backward algorithm, which is implemented by the SRI language modeling toolkit (Stolcke, 2002).

In practice, because of data sparsity, the language model in Equation 1 must be modified to include smoothing for unseen phoneme sequences. We use a backoff M-gram model with Witten-Bell discounting (Witten and Bell, 1991). We set the phoneme language model's order to $M = 5$, which gave good results in previous work (Yu and Seide, 2005).

3 Incorporating Query Degradations

One problem with the generative approach is that recognition error is not modeled (apart from the uncertainty captured in the phoneme lattice). The essential problem is that while the method hopes to model $P(Q|\mathcal{L})$, it is in fact only able to model the probability of one *degradation* H in the lattice, that is $P(H|\mathcal{L})$. We define a query degradation as any phoneme sequence (including the lexical sequence) which may, with some estimated probability, occur in an errorful phonemic representation of the audio (either a one-best or lattice hypothesis). Because of speaker variation and because recognition is errorful, we ought to also consider non-lexical degradations of the query phoneme sequence. That is, we should incorporate $P(H|Q)$ in our ranking function.

It has previously been demonstrated that allowing for phoneme confusability can significantly increase spoken term detection performance on one-best phoneme transcripts (Chaudhari and Picheny, 2007; Schone et al., 2005) and in phonemic lattices (Foote et al., 1997). These methods work by allowing weighted substitution costs in minimum-edit-distance matching. Previously, these substitution costs have been maximum-likelihood estimates of $P(H|Q)$ for each phoneme, where $P(H|Q)$ is easily computed from a phoneme confusion matrix after aligning the reference and one-best hypothesis transcript under a minimum edit distance criterion. Similar methods have also been used in other language processing applications. For example, in (Kolak, 2005), one-for-one character substitutions, insertions and deletions were considered in a generative model of errors in OCR.

In this work, because we are focused on constructing inverted indices of audio files (for speed and to conserve space), we must generalize our method of incorporating query degradations in the ranking function. Given a degradation model $P(H|Q)$, we take as our ranking function the expectation of the generative baseline estimate $N_{\mathcal{L}} \cdot \hat{P}(H|\mathcal{L})$ with respect to $P(H|Q)$,

$$\widehat{tf}_G(Q, \mathcal{L}) = \sum_{H \in \mathcal{H}} \left[\hat{P}(H|\mathcal{L}) \cdot N_{\mathcal{L}}\right] \cdot P(H|Q), \quad (2)$$

where $\mathcal{H}$ is the set of degradations. Note that, while we consider the expected value of our baseline term frequency estimator with respect to $P(H|Q)$, this general approach could be used with any other term frequency estimator.

Our formulation is similar to approaches taken in OCR document retrieval, using degradations of character sequences (Darwish and Magdy, 2007; Darwish, 2003). For vocabulary-independent spoken term detection, perhaps the most closely related formulation is provided by (Mamou and Ramabhadran, 2008). In that work, they ranked utterances by the weighted average of their matching score, where the weights were confidences from a grapheme to phoneme system's first several hypotheses for a word's pronunciation. The matching scores were edit distances, where substitution costs were weighted using phoneme confusability. Accordingly, their formulation was not aimed at accounting for errors in recognition per se, but rather for errors in hypothesizing pronunciations. We expect this accounts for their lack of significant improvement using the method.

Since we don't want to sum over all possible recognition hypotheses H, we might instead sum over the smallest set $\mathcal{H}$ such that $\sum_{H \in \mathcal{H}} P(H|Q) \geq \gamma$. That is, we could take the most probable degradations until their cumulative probability exceeds some threshold γ. In practice, however, because

degradation probabilities can be poorly scaled, we instead take a fixed number of degradations and normalize their scores. When a query is issued, we apply a degradation model to learn the top few phoneme sequences $\mathcal{H}$ that are most likely to have been recognized, under the model. In the machine translation literature, this process is commonly referred to as *decoding*.

We now turn to the modeling of query degradations H given a phoneme sequence Q, $P(H|Q)$. First, we consider a simple baseline approach in Section 3.1. Then, in Section 3.2, we propose a more powerful technique, using state-of-the-art machine translation methods to hypothesize our degradations.

3.1 Baseline Query Degradations

Schone et al. (2005) used phoneme confusion matrices created by aligning hypothesized and reference phoneme transcripts to weight edit costs for a minimum-edit distance based search in a one-best phoneme transcript. Foote et al. (1997) had previously used phoneme lattices, although with *ad hoc* edit costs and without efficient indexing. In this work, we do not want to linearly scan each phoneme lattice for our query's phoneme sequence, preferring instead to look up sequences in the inverted indices containing phoneme sequences.

Our baseline degradation approach is related to the edit-cost approach taken by (Schone et al., 2005), although we generalize it so that it may be applied within Equation 2 and we consider speech recognition hypotheses beyond the one-best hypothesis. First, we randomly generate N traversals of each phonemic recognition lattice. These traversals are random paths through the lattice (i.e., we start at the beginning of the lattice and move to the next node, where our choice is weighted by the outgoing arcs' probabilities). Then, we align each of these traversals with its reference transcript using a minimum-edit distance criterion. Phone confusion matrices are then tabulated from the aggregated insertion, substitution, and deletion counts across all traversals of all lattices. From these confusion matrices, we compute unsmoothed estimates of $P(h|r)$, the probability of a phoneme h being hypothesized given a reference phoneme r.

Making an independence assumption, our baseline degradation model for a query with m phonemes is then $P(H|Q) = \prod_{i=1}^{m} P(h_i|r_i)$. We efficiently compute the most probable degradations for a query Q using a lattice of possible degradations and the forward backward algorithm. We call this baseline degradation approach CMQD (Confusion Matrix based Query Degradation).

AY	K	M	AA	N
Vowel	Consonant	Semi-vowel	Vowel	Semi-vowel
Dipthong	Voiceless plosive	Nasal	Back vowel	Nasal

Figure 1: Three levels of annotation used by the factored phrase-based query degradation model.

3.2 Phrase-Based Query Degradation

One problem with CMQD is that we only allow insertions, deletions, and one-for-one substitutions. It may be, however, that certain pairs of phonemes are commonly hypothesized for a particular reference phoneme (in the language of statistical machine translation, we might say that we should allow some non-zero *fertility*). Second, there is nothing to discourage query degradations which are unlikely under an (errorful) language model—that is, degradations that are not observed in the speech hypotheses. Finally, CMQD doesn't account for similarities between phoneme classes. While some of these deficiencies could be addressed with an extension to CMQD (e.g., by expanding the degradation lattices to include language model scores), we can do better using a more powerful modeling framework. In particular, we adopt the approach of phrase-based statistical machine translation (Koehn et al., 2003; Koehn and Hoang, 2007). This approach allows for multiple-phoneme to multiple-phoneme substitutions, as well as the soft incorporation of additional linguistic knowledge (e.g., phoneme classes). This is related to previous work allowing higher order phoneme confusions in bigram or trigram contexts (Chaudhari and Picheny, 2007), although they used a fuzzy edit distance measure and did not incorporate other evidence in their model (e.g., the phoneme language model score). The reader is referred to (Koehn and Hoang, 2007; Koehn et al., 2007) for detailed information about phrase-based statistical machine translation. We give a brief outline here, sufficient only to provide background for our query degradation application.

Statistical machine translation systems work by

converting a source-language sentence into the most probable target-language sentence, under a model whose parameters are estimated using example sentence pairs. Phrase-based machine translation is one variant of this statistical approach, wherein multiple-word *phrases* rather than isolated words are the basic translation unit. These phrases are generally not linguistically motivated, but rather learned from co-occurrences in the paired example translation sentences. We apply the same machinery to hypothesize our pronunciation degradations, where we now translate from the "source-language" reference phoneme sequence Q to the hypothesized "target-language" phoneme sequence H.

Phrase-based translation is based on the noisy channel model, where Bayes rule is used to reformulate the translation probability for translating a reference query Q into a hypothesized phoneme sequence H as

$$\arg\max_H P(H|Q) = \arg\max_H P(Q|H)P(H).$$

Here, for example, $P(H)$ is the language model probability of a degradation H and $P(Q|H)$ is the conditional probability of the reference sequence Q given H. More generally however, we can incorporate other *feature functions* of H and Q, $h_i(H, Q)$, and with varying weights. This is implemented using a log-linear model for $P(H|Q)$, where the model covariates are the functions $h_i(H, Q)$, so that

$$P(H|Q) = \frac{1}{Z} \exp \sum_{i=1}^{n} \lambda_i h_i(H, Q)$$

The parameters λ_i are estimated by MLE and the normalizing Z need not be computed (because we will take the argmax). Example feature functions include the language model probability of the hypothesis and a hypothesis length penalty.

In addition to feature functions being defined on the surface level of the phonemes, they may also be defined on non-surface annotation levels, called *factors*. In a word translation setting, the intuition is that statistics from morphological variants of a lexical form ought to contribute to statistics for other variants. For example, if we have never seen the word *houses* in language model training, but have examples of *house*, we still can expect *houses are* to be more probable than *houses fly*. In other words, factors allow us to collect improved statistics on sparse data. While sparsity might appear to be less of a problem for phoneme degradation modeling (because the token inventory is comparatively very small), we nevertheless may benefit from this approach, particularly because we expect to rely on higher order language models and because we have rather little training data: only 22,810 transcribed utterances (about 600k reference phonemes).

In our case, we use two additional annotation layers, based on a simple grouping of phonemes into broad classes. We consider the phoneme itself, the broad distinction of vowel and consonant, and a finer grained set of classes (e.g., front vowels, central vowels, voiceless and voiced fricatives). Figure 1 shows the three annotation layers we consider for an example reference phoneme sequence. After mapping the reference and hypothesized phonemes to each of these additional factor levels, we train language models on each of the three factor levels of the hypothesized phonemes. The language models for each of these factor levels are then incorporated as features in the translation model.

We use the open source toolkit `Moses` (Koehn et al., 2007) as our phrase-based machine translation system. We used the SRI language modeling toolkit to estimate interpolated 5-gram language models (for each factor level), and smoothed our estimates with Witten-Bell discounting (Witten and Bell, 1991). We used the default parameter settings for `Moses`'s training, with the exception of modifying `GIZA++`'s default maximum fertility from 10 to 4 (since we don't expect one reference phoneme to align to 10 degraded phonemes). We used default decoding settings, apart from setting the distortion penalty to prevent any reorderings (since alignments are logically constrained to never cross). For the rest of this chapter, we refer to our phrase-based query degradation model as PBQD. We denote the phrase-based model using factors as PBQD-Fac.

Figure 2 shows an example alignment learned for a reference and one-best phonemic transcript. The reference utterance "snow white and the seven dwarves" is recognized (approximately) as "no white a the second walks". Note that the phrase-based system is learning not only acoustically plausible confusions, but critically, also confusions aris-

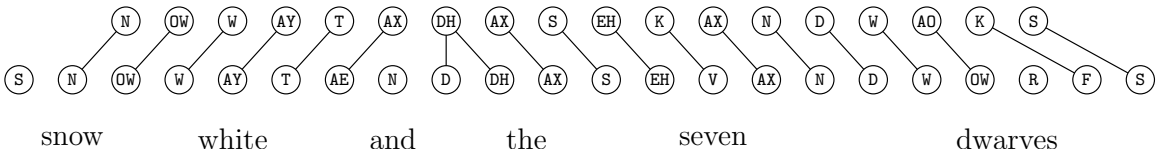


Figure 2: An alignment of hypothesized and reference phoneme transcripts from the multigram phoneme recognizer, for the phrase-based query degradation model.

ing from the phonemic recognition system's peculiar construction. For example, while V and K may not be acoustically similar, they are still confusable—within the context of S EH—because multigram language model data has many examples of the word *second*. Moreover, while the word *dwarves* (D-W-OW-R-F-S) is not present in the dictionary, the words *dwarf* (D-W-AO-R-F) and *dwarfed* (D-W-AO-R-F-T) are present (*N.B.*, the change of vowel from AO to OW between the OOV and in vocabulary pronunciations). While CMQD would have to allow a deletion and two substitutions (without any context) to obtain the correct degradation, the phrase-based system can align the complete phrase pair from training and exploit context. Here, for example, it is highly probable that the errorfully hypothesized phonemes W AO will be followed by *K*, because of the prevalence of *walk* in language model data.

4 Experiments

An appropriate and commonly used measure for RUR is Mean Average Precision (MAP). Given a ranked list of utterances being searched through, we define the *precision* at position i in the list as the proportion of the top i utterances which actually contain the corresponding query word. Average Precision (AP) is the average of the precision values computed for each position containing a relevant utterance. To assess the effectiveness of a system across multiple queries, Mean Average Precision is defined as the arithmetic mean of per-query average precision, $\mathrm{MAP} = \frac{1}{n}\sum_n \mathrm{AP}_n$. Throughout this paper, when we report statistically significant improvements in MAP, we are comparing AP for paired queries using a Wilcoxon signed rank test at $\alpha = 0.05$.

Note, RUR is different than spoken term detection in two ways, and thus warrants an evaluation measure (e.g., MAP) different than standard spoken term detection measures (such as NIST's *actual term weighted value* (Fiscus et al., 2006)). First, STD measures require locating a term with granularity finer than that of an utterance. Second, STD measures are computed using a fixed detection threshold. This latter requirement will be unnecessary in many applications (e.g., where a user might prefer to decide themselves when to stop reading down the ranked list of retrieved utterances) and unlikely to be helpful for downstream evidence combination (where we may prefer to keep all putative hits and weight them by some measure of confidence).

For our evaluation, we consider retrieving short utterances from seventeen fully transcribed MALACH interviews. Our query set contains all single words occurring in these interviews that are OOV with respect to the word dictionary. This gives us a total of 261 query terms for evaluation. Note, query words are also not present in the multigram training transcripts, in any language model training data, or in any transcripts used for degradation modeling. Some example query words include BUCHENWALD, KINDERTRANSPORT, and SONDERKOMMANDO.

To train our degradation models, we used a held out set of 22,810 manually transcribed utterances. We run each recognition system (phoneme, multigram, and word) on these utterances and, for each, train separate degradation models using the aligned reference and hypothesis transcripts. For CMQD, we computed 100 random traversals on each lattice, giving us a total of 2,281,000 hypothesis and reference pairs to align for our confusion matrices.

5 Results

We first consider an intrinsic measure of the three speech recognition systems we consider, namely Phoneme Error Rate (PER). Phoneme Error Rate is calculated by first producing an alignment of

the hypothesis and reference phoneme transcripts. The counts of each error type are used to compute $PER = 100 \cdot \frac{S+D+I}{N}$, where S, D, I are the number of substitutions, insertions, and deletions respectively, while N is the phoneme length of the reference. Results are shown in Table 1. First, we see that the PER for the multigram system is roughly half that of the phoneme-only system. Second, we find that the word system achieves a considerably lower PER than the multigram system. We note, however, that since these are not true phonemes (but rather phonemes copied over from pronunciation dictionaries and word transcripts), we must cautiously interpret these results. In particular, it seems reasonable that this framework will overestimate the strength of the word based system. For comparison, on the same train/test partition, our word-level system had a *word* error rate of 31.63. Note, however, that automatic word transcripts can not contain our OOV query words, so word error rate is reported only to give a sense of the difficulty of the recognition task.

Table 1 shows our baseline RUR evaluation results. First, we find that the generative model yields statistically significantly higher MAP using words or multigrams than phonemes. This is almost certainly due to the considerably improved phoneme recognition afforded by longer recognition units. Second, many more unique phoneme sequences typically occur in phoneme lattices than in their word or multigram counterparts. We expect this will increase the false alarm rate for the phoneme system, thus decreasing MAP.

Surprisingly, while the word-based recognition system achieved considerably lower phoneme error rates than the multigram system (see Table 1), the word-based generative model was in fact indistinguishable from the same model using multigrams. We speculate that this is because the method, as it is essentially a language modeling approach, is sensitive to data sparsity and requires appropriate smoothing. Because multigram lattices incorporate smaller recognition units, which are not constrained to be English words, they naturally produce smoother phoneme language models than a word-based system. On the other hand, the multigram system is also not statistically significantly better than the word-based generative model, suggesting this may be a promising area for future work.

Table 1 shows results using our degradation models. Query degradation appears to help all systems with respect to the generative baseline. This agrees with our intuition that, for RUR, low MAP on OOV terms is predominately driven by low recall.[1] Note that, at one degradation, CMQD has the same MAP as the generative model, since the most probable degradation under CMQD is almost always the reference phoneme sequence. Because the CMQD model can easily hypothesize implausible degradations, we see the MAP increases modestly with a few degradations, but then MAP decreases. In contrast, the MAP of the phrase-based system (PBQD-Fac) increases through to 500 query degradations using multigrams. The phonemic system appears to achieve its peak MAP with fewer degradations, but also has a considerably lower best value.

The non-factored phrase-based system PBQD achieves a peak MAP considerably larger than the peak CMQD approach. And, likewise, using additional factor levels (PBQD-Fac) also considerably improves performance. Note especially that, using multiple factor levels, we not only achieve a higher MAP, but also a higher MAP when only a few degradations are possible.

To account for errors in phonemic recognition, we have taken two steps. First, we used longer recognition units which we found significantly improved MAP while using our baseline RUR technique. As a second method for handling recognition errors, we also considered variants of our ranking function. In particular, we incorporated query degradations hypothesized using factored phrase-based machine translation. Comparing the MAP for PBQD-Fac with MAP using the generative baseline for the most improved indexing system (the word system), we find that this degradation approach again statistically significantly improved MAP. That is, these two strategies for handling recognition errors in RUR appear to work well in combination.

Although we focused on vocabulary-independent RUR, downstream tasks such as *ad hoc* speech retrieval will also want to incorporate evidence from in-vocabulary query words. This makes

[1] We note however that the preferred operating point in the tradeoff between precision and recall will be task specific. For example, it is known that precision errors become increasingly important as collection size grows (Shao et al., 2008).

Method	Phone Source	PER	QD Model	Baseline	Query Degradations			
					1	5	50	500
Degraded Model	Phonemes	64.4	PBQD-Fac	0.0387	0.0479	0.0581	**0.0614**	0.0612
	Multigrams	32.1	CMQD	0.1258	0.1258	0.1272	0.1158	0.0991
	Multigrams	32.1	PBQD	0.1258	0.1160	0.1283	0.1347	0.1317
	Multigrams	32.1	PBQD-Fac	0.1258	0.1238	0.1399	0.1510	**0.1527**
	Words	20.5	PBQD-Fac	0.1255	0.1162	0.1509	**0.1787**	0.1753

Table 1: PER and MAP results for baseline and degradation models. The best result for each indexing approach is shown in bold.

our query degradation approach which indexed phonemes from word-based LVCSR particularly attractive. Not only did it achieve the best MAP in our evaluation, but this approach also allows us to construct recognition lattices for both in and out-of-vocabulary query words without running a second, costly, recognition step.

6 Conclusion

Our goal in this work was to rank utterances by our confidence that they contained a previously unseen query word. We proposed a new approach to this task using hypothesized degradations of the query word's phoneme sequence, which we produced using a factored phrase-based machine translation model. This approach was principally motivated by the mismatch between the query's phonemes and the recognition phoneme sequences due to errorful speech indexing. Our approach was constructed and evaluated using phoneme-, multigram-, and word-based indexing, and significant improvements in MAP using each indexing system were achieved. Critically, these significant improvements were in addition to the significant gains we achieved by constructing our index with longer recognition units.

While PBQD-Fac outperformed CMQD averaging over all queries in our evaluation, as expected, there may be particular query words for which this is not the case. Table 2 shows example degradations using both the CMQD and PBQD-Fac degradation models for multigrams. The query word is *Mengele*. We see that CMQD degradations are near (in an edit distance sense) to the reference pronunciation (`M-EH-NX-EY-L-EH`), while the phrase-based degradations tend to sound like commonly occurring words (*mental, meant a lot, men they. . . , mentally*). In this case, the lexical phoneme sequence does not occur in the PBQD-Fac degradations until degradation nineteen. Because deleting `EH` has the same cost irrespective of context for CMQD, both CMQD degradations 2 and 3 are given the same pronunciation weight. Here, CMQD performs considerably better, achieving an average precision of 0.1707, while PBQD-Fac obtains only 0.0300. This suggests that occasionally the phrase-based language model may exert too much influence on the degradations, which is likely to increase the incidence of false alarms. One solution, for future work, might be to incorporate a false alarm model (e.g., down-weighting putative occurrences which look suspiciously like non-query words). Second, we might consider training the degradation model in a discriminative framework (e.g., training to optimize a measure that will penalize degradations which cause false alarms, even if they are good candidates from the perspective of MLE). We hope that the ideas presented in this paper will provide a solid foundation for this future work.

CMQD	Phrase-based
`M-EH-NX-EY-L-EH`	`M-EH-N-T-AX-L`
`M-EH-NX-EY-L`	`M-EH-N-T-AX-L-AA-T`
`M-NX-EY-L-EH`	`AH-AH-AH-AH-M-EH-N-T-AX-L`
`M-EH-NX-EY-EH`	`M-EH-N-DH-EY-L-EH`
`M-EH-NX-L-EH`	`M-EH-N-T-AX-L-IY`

Table 2: The top five degradations and associated probabilities using the CMQD and PBQD-Fac models, for the term *Mengele* using multigram indexing.

References

W. Byrne et al. 2004. Automatic Recognition of Spontaneous Speech for Access to Multilingual Oral History Archives. *IEEE Transactions on Speech and Audio Processing, Special Issue on Spontaneous Speech Processing*, 12(4):420–435, July.

U.V. Chaudhari and M. Picheny. 2007. Improvements in phone based audio search via constrained match with high order confusion estimates. *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 665–670, Dec.

Kareem Darwish and Walid Magdy. 2007. Error correction vs. query garbling for Arabic OCR document retrieval. *ACM Trans. Inf. Syst.*, 26(1):5.

Kareem M. Darwish. 2003. *Probabilistic Methods for Searching OCR-Degraded Arabic Text*. Ph.D. thesis, University of Maryland, College Park, MD, USA. Directed by Bruce Jacob and Douglas W. Oard.

S. Deligne and F. Bimbot. 1997. Inference of Variable-length Acoustic Units for Continuous Speech Recognition. In *ICASSP '97: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1731–1734, Munich, Germany.

Jonathan Fiscus et al. 2006. English Spoken Term Detection 2006 Results. In *Presentation at NIST's 2006 STD Eval Workshop*.

J.T. Foote et al. 1997. Unconstrained keyword spotting using phone lattices with application to spoken document retrieval. *Computer Speech and Language*, 11:207—224.

Philipp Koehn and Hieu Hoang. 2007. Factored Translation Models. In *EMNLP '07: Conference on Empirical Methods in Natural Language Processing*, June.

Philipp Koehn et al. 2003. Statistical phrase-based translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Morristown, NJ, USA. Association for Computational Linguistics.

Philipp Koehn et al. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL '07: Proceedings of the 2007 Conference of the Association for Computational Linguistics, demonstration session*, June.

Okan Kolak. 2005. *Rapid Resource Transfer for Multilingual Natural Language Processing*. Ph.D. thesis, University of Maryland, College Park, MD, USA. Directed by Philip Resnik.

Jonathan Mamou and Bhuvana Ramabhadran. 2008. Phonetic Query Expansion for Spoken Document Retrieval. In *Interspeech '08: Conference of the International Speech Communication Association*.

Spyros Matsoukas et al. 2005. The 2004 BBN 1xRT Recognition Systems for English Broadcast News and Conversational Telephone Speech. In *Interspeech '05: Conference of the International Speech Communication Association*, pages 1641–1644.

K. Ng and V.W. Zue. 2000. Subword-based approaches for spoken document retrieval. *Speech Commun.*, 32(3):157–186.

J. Scott Olsson. 2008a. Combining Speech Retrieval Results with Generalized Additive Models. In *ACL '08: Proceedings of the 2008 Conference of the Association for Computational Linguistics*.

J. Scott Olsson. 2008b. Vocabulary Independent Discriminative Term Frequency Estimation. In *Interspeech '08: Conference of the International Speech Communication Association*.

Pavel Pecina, Petra Hoffmannova, Gareth J.F. Jones, Jianqiang Wang, and Douglas W. Oard. 2007. Overview of the CLEF-2007 Cross-Language Speech Retrieval Track. In *Proceedings of the CLEF 2007 Workshop on Cross-Language Information Retrieval and Evaluation*, September.

R. Prasad et al. 2005. The 2004 BBN/LIMSI 20xRT English Conversational Telephone Speech Recognition System. In *Interspeech '05: Conference of the International Speech Communication Association*.

S.E. Robertson. 1977. The Probability Ranking Principle in IR. *Journal of Documentation*, pages 281–286.

M. Saraclar and R. Sproat. 2004. Lattice-Based Search for Spoken Utterance Retrieval. In *NAACL '04: Proceedings of the 2004 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*.

P. Schone et al. 2005. Searching Conversational Telephone Speech in Any of the World's Languages.

Jian Shao et al. 2008. Towards Vocabulary-Independent Speech Indexing for Large-Scale Repositories. In *Interspeech '08: Conference of the International Speech Communication Association*.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *ICSLP '02: Proceedings of 2002 International Conference on Spoken Language Processing*.

I. H. Witten and T. C. Bell. 1991. The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression. *IEEE Trans. Information Theory*, 37(4):1085–1094.

Peng Yu and Frank Seide. 2005. Fast Two-Stage Vocabulary-Independent Search In Spontaneous Speech. In *ICASSP '05: Proceedings of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing*.

P. Yu et al. Sept. 2005. Vocabulary-Independent Indexing of Spontaneous Speech. *IEEE Transactions on Speech and Audio Processing*, 13(5):635–643.

Japanese Query Alteration Based on Semantic Similarity

Masato Hagiwara
Nagoya University
Furo-cho, Chikusa-ku
Nagoya 464-8603, Japan
hagiwara@kl.i.is.nagoya-u.ac.jp

Hisami Suzuki
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
hisamis@microsoft.com

Abstract

We propose a unified approach to web search query alterations in Japanese that is not limited to particular character types or orthographic similarity between a query and its alteration candidate. Our model is based on previous work on English query correction, but makes some crucial improvements: (1) we augment the query-candidate list to include orthographically dissimilar but semantically similar pairs; and (2) we use kernel-based lexical semantic similarity to avoid the problem of data sparseness in computing query-candidate similarity. We also propose an efficient method for generating query-candidate pairs for model training and testing. We show that the proposed method achieves about 80% accuracy on the query alteration task, improving over previously proposed methods that use semantic similarity.

1 Introduction

Web search query correction is an important problem to solve for robust information retrieval given how pervasive errors are in search queries: it is said that more than 10% of web search queries contain errors (Cucerzan and Brill, 2004). English query correction has been an area of active research in recent years, building on previous work on general-purpose spelling correction. However, there has been little investigation of query correction in languages other than English.

In this paper, we address the issue of query correction, and more generally, query alteration in Japanese. Japanese poses particular challenges to the query correction task due to its complex writing system, summarized in Fig. 1[1]. There are four main character types, including two types of *kana* (phonetic alphabet - *hiragana* and *katakana*), *kanji* (ideographic - characters represent meaning) and Roman alphabet; a word can be legitimately spelled in multiple ways, combining any of these character sets. For example, the word for 'protein' can be spelled as たんぱくしつ (all in hiragana), タンパク質 (katakana+kanji), 蛋白質 (all in kanji) or たん白質 (hiragana+kanji), all pronounced in the same way (tanpakushitsu). Some examples of these spelling variants are shown in Fig. 1 with the prefix Sp: as is observed from the figure, spelling variation occurs within and across different character types. Resolving these variants will be essential not only for information retrieval but practically for all NLP tasks.

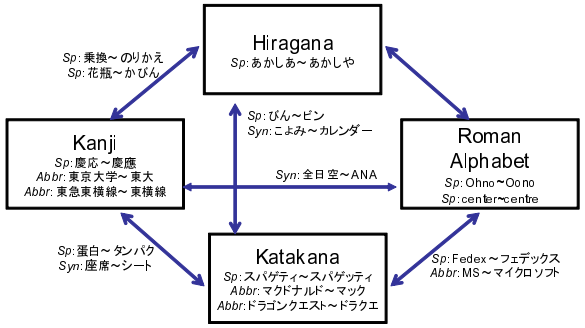


Figure 1: Japanese character types and spelling variants

A particularly prolific source of spelling variations in Japanese is katakana. Katakana characters are used to transliterate words from English and other foreign languages, and as such, the variations in the source language pronunciation as well as the ambiguity in sound adaptation are reflected in the katakana spelling. For example, Masuyama et al. (2004) report that at least six distinct transliterations of the word 'spaghetti' (スパゲッティ, スパゲティー, etc.) are attested in the newspaper corpus they studied. Normalizing katakana spelling variations has been the subject of research by itself (Aramaki et al., 2008; Masuyama et al., 2004). Similarly, English-to-katakana transliteration (e.g., 'fedex' as フェデックス fedekkusu in Fig. 1) and katakana-to-

[1]The figure is somewhat over-simplified as it does not include any word consisting of multiple character types. It also does not include examples of spelling mistakes and variants in word segmentation.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 191–199,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

English back-transliteration (e.g., フェデックス back into 'fedex') have also been studied extensively (Bilac and Tanaka, 2004; Brill et al., 2001; Knight and Graehl, 1998), as it is an essential component for machine translation. To our knowledge, however, there has been no work that addresses spelling variation in Japanese generally.

In this paper, we propose a general approach to query correction/alteration in Japanese. Our goal is to find precise re-write candidates for a query, be it a correction of a spelling error, normalization of a spelling variant, or finding a strict synonym including abbreviations (e.g., MS　マイクロソフト 'Microsoft', prefixed by Abbr in Fig. 1) and true synonyms (e.g., 座席 (translation of 'seat')　シート (transliteration of 'seat', indicated by *Syn* in Fig. 1)[2]. Our method is based on previous work on English query correction in that we use both spelling and semantic similarity between a query and its alteration candidate, but is more general in that we include alteration candidates that are not similar to the original query in spelling. In computing semantic similarity, we adopt a kernel-based method (Kandola et al., 2002), which improves the accuracy of the query alteration results over previously proposed methods. We also introduce a novel approach to creating a dataset of query and alteration candidate pairs efficiently and reliably from query session logs.

2 Related Work

The key difference between traditional general-purpose spelling correction and search query correction lies in the fact that the latter cannot rely on a lexicon: web queries are replete with valid out-of-dictionary words which are not mis-spellings of in-vocabulary words. Cucerzan and Brill (2004) pioneered the research of query spelling correction, with an excellent description of how a traditional dictionary-based speller had to be adapted to solve the realistic query correction problem. The model they proposed is a source-channel model, where the source model is a word bigram model trained on query logs, and the channel model is based on a weighted Damerau-Levenshtein edit distance. Brill and Moore (2000) proposed a general, improved source model for general spelling correction, while Ahmad and Kondrak (2005) learned a spelling error model from search query logs using the Expectation Maximization algorithm, without relying on a training set of misspelled words and their corrections.

Extending the work of Cucerzan and Brill (2004), Li et al. (2006) proposed to include semantic similarity between the query and its correction candidate. They point out that *adventura* is a common misspelling of *aventura*, not *adventure*, and this cannot be captured by a simple string edit distance, but requires some knowledge of distributional similarity. Distributional similarity is measured by the similarity of the context shared by two terms, and has been successfully applied to many natural language processing tasks, including semantic knowledge acquisition (Lin, 1998).

Though the use of distributional similarity improved the query correction results in Li et al.'s work, one problem is that it is sparse and is not available for many rarer query strings. Chen et al. (2007) addressed this problem by using external information (i.e., web search results); we take a different approach to solve the sparseness problem, namely by using semantic kernels.

Jones et al. (2006a) generated Japanese query alteration pairs from by mining query logs and built a regression model which predicts the quality of query rewriting pairs. Their model includes a wide variety of orthographical features, but not semantic similarity features.

3 Query Alteration Model

3.1 Problem Formulation

We employ a formulation of query alteration model that is similar to conventional query correction models. Given a query string q as input, a query correction model finds a correct alteration c^* within the confusion set of q, so that it maximizes the posterior probability:

$$c^* = \arg \max_{c \in \mathrm{CF}(q) \subset C} P(c|q) \quad (1)$$

where C is the set of all white-space separated words and their bigrams in query logs in our case[3], and

[2] Our goal is to harvest alternation candidates; therefore, exactly how they are used in the search task (whether it is used to *substitute* the original query, to *expand* it, or simply to *suggest* an alternative) is not a concern to us here.

[3] In regular text, Japanese uses no white spaces to separate words; however, white spaces are often (but not consistently)

$\mathrm{CF}(q) \subset C$ is the confusion set of q, consisting of the candidates within a certain edit distance from q, i.e., $\mathrm{CF}(q) = \{c \in C | \mathrm{ED}(q, c) < \theta\}$. We set $\theta = 24$ using an unnormalized edit distance. The detail of the edit distance $\mathrm{ED}(q, c)$ is described in Section 3.2. The query string q itself is contained in $\mathrm{CF}(q)$, and if the model output is different from q, it means the model suggests a query alteration. Formulated in this way, both query error detection and alteration are performed in a unified way.

After computing the posterior probability of each candidate in $\mathrm{CF}(q)$ by the source channel model (Section 3.2), an N-best list is obtained as the initial candidate set C_0, which is then augmented by the bootstrapping method *Tchai* (Section 3.4) to create the final candidate list $C(q)$. The candidates in $C(q)$ are re-ranked by a maximum entropy model (Section 3.5) and the candidate with the highest posterior probability is selected as the output.

3.2 Source Channel Model

Source channel models are widely used for spelling and query correction (Brill and Moore, 2000; Cucerzan and Brill, 2004). Instead of directly computing Eq. (1), we can decompose the posterior probability using Bayes' rule as:

$$c^* = \arg \max_{c \in \mathrm{CF}(q) \subset C} P(c)P(q|c), \tag{2}$$

where the source model $P(c)$ measures how probable the candidate c is, while the error model $P(q|c)$ measures how similar q and c are.

For the source model, an n-gram based statistical language model is the standard in previous work (Ahmad and Kondrak, 2005; Li et al., 2006). Word n-gram models are simple to create for English, which is easy to tokenize and to obtain word-based statistics, but this is not the case with Japanese. Therefore, we simply considered the whole input string as a candidate to be altered, and used the relative frequency of candidates in the query logs to build the language model:

$$P(c) = \frac{\mathrm{Freq(c)}}{\sum_{c' \in C} \mathrm{Freq(c')}}. \tag{3}$$

For the error model, we used an improved channel model described in (Brill and Moore, 2000), which we call the *alpha-beta model* in this paper. The model is a weighted extension of the normal Damerau-Levenshtein edit distance which equally penalizes single character insertion, substitution, or deletion operations (Damerau, 1964; Levenshtein, 1966), and considers generic edit operations of the form $\alpha \rightarrow \beta$, where α and β are any (possibly null) strings. From misspelled/correct word pairs, *alpha-beta* trains the probability $P(\alpha \rightarrow \beta|\mathrm{PSN})$, conditioned by the position PSN of α in a word, where PSN $\in$ {start of word, middle of word, end of word}. Under this model, the probability of rewriting a string w to a string s is calculated as:

$$P_{\alpha\beta}(s|w) = \max_{R \in \mathrm{Part}(w), T \in \mathrm{Part}(s)} \prod_{i=1}^{|R|} P(R_i \rightarrow T_i|\mathrm{PSN}(R_i)),$$

which corresponds to finding best partitions R and T in all possible partitions $\mathrm{Part}(w)$ and $\mathrm{Part}(s)$. Brill and Moore (2000) reported that this model gave a significant improvement over conventional edit distance methods.

Brill et al. (2001) applied this model for extracting katakana-English transliteration pairs from query logs. They trained the edit distance between character chunks of katakana and Roman alphabets, after converting katakana strings to Roman script. We also trained this model using 59,754 katakana-English pairs extracted from aligned Japanese and English Wikipedia article titles. In this paper we allowed $|\alpha|, |\beta| \leq 3$. The resulting edit distance is obtained as the negative logarithm of the *alpha-beta* probability, i.e., $\mathrm{ED}_{\alpha\beta}(q|c) = -\log P_{\alpha\beta}(q|c)$.

Since the edit operations are directional and c and q can be any string consisting of katakana and English, distance in both directions were considered. We also included a modified edit distance $\mathrm{ED}_{\mathrm{hd}}$ for simple kana-kana variations after converting them into Roman script. The distance $\mathrm{ED}_{\mathrm{hd}}$ is essentially the same as the normal Damerau-Levenshtein edit distance, with the modification that it does not penalize character halving ($aa \rightarrow a$) and doubling ($a \rightarrow aa$), because a large part of katakana variants only differ in halving/doubling (e.g. スパゲティ (`supageti`) vs スパゲティー (`supagetii`)[4].

The final error probability is obtained from the minimum of these three distances:

used to separate words in Japanese search queries, due to their keyword-based nature.

[4]However, character length can be distinctive in katakana, as in ビル `biru` 'building' vs. ビール `biiru` 'beer'.

$$\mathrm{ED}(q,c) = \min[\mathrm{ED}_{\alpha\beta}(q|c), \mathrm{ED}_{\alpha\beta}(c|q), \mathrm{ED}_{\mathrm{hd}}(q,c)], \quad (4)$$

$$P(q|c) = \exp[-\mathrm{ED}(q,c)] \quad (5)$$

where every edit distance is normalized to [0, 1] by multiplying by a factor of $2/(|q||c|)$ so that it does not depend on the length of the input strings[5].

3.3 Kernel-based Lexical Semantic Similarity

3.3.1 Distributional Similarity

The source channel model described in Section 3.2 only considers language and error models and cannot capture semantic similarity between the query and its correction candidate. To address this issue, we use distributional similarity (Lin, 1998) estimated from query logs as additional evidence for query alteration, following Li et al. (2006).

For English, it is relatively easy to define the context of a word based on the bag-of-words model. As this is not expected to work on Japanese, we define context as everything but the query string in a query log, as Paşca et al. (2006) and Komachi and Suzuki (2008) did for their information extraction tasks. This formulation does not involve any segmentation or boundary detection, which makes this method fast and robust. On the other hand, this may cause additional sparseness in the vector representation; we address this issue in the next two sections.

Once the context of a candidate c_i is defined as the patterns that the candidate co-occurs with, it can be represented as a vector $\mathbf{c}_i = [\mathrm{pmi}(c_i, p_1), \ldots, \mathrm{pmi}(c_i, p_M)]'$, where M denotes the number of context patterns and x' is the transposition of a vector (or possibly a matrix) x. The elements of the vector are given by pointwise mutual information between the candidate c_i and the pattern p_j, computed as:

$$\mathrm{pmi}(c_i, p_j) = \log \frac{|c_i, p_j|}{|c_i, *||*, p_j|}, \quad (6)$$

where $|c_i, p_j|$ is the frequency of the pattern p_j instantiated with the candidate c_i, and '*' denotes a wildcard, i.e., $|c_i, *| = \sum_p |c_i, p|$ and $|*, p_j| = \sum_c |c, p_j|$. With these defined, the distributional similarity can be calculated as cosine similarity. Let $\hat{\mathbf{c}}_i$ be the L2-normalized pattern vector of the candidate c_i, and $X = \{\hat{\mathbf{c}}_i\}$ be the candidate-pattern co-occurrence matrix. The candidate similarity matrix K can then be obtained as $K = X'X$. In the following, the (i, j)-element of the matrix K is denoted as K_{ij}, which corresponds to the cosine similarity between candidates c_i and c_j.

3.3.2 Semantic Kernels

Although distributional similarity serves as strong evidence for semantically relevant candidates, directly applying the technique to query logs faces the sparseness problem. Because context patterns are drawn from query logs and can also contain spelling errors, alterations, and word permutations as much as queries do, context differs so greatly in representations that even related candidates might not have sufficient contextual overlap between them. For example, a candidate "YouTube" matched against the patterns "YouTube+movie", "movie+YouTube" and "You-Tube+movii" (with a minor spelling error) will yield three distinct patterns "#+movie", "movie+#" and "#+movii"[6], which will be treated as three separate dimensions in the vector space model.

This sparseness problem can be partially addressed by considering the correlation between patterns. Kandola et al. (2002) proposed new kernel-based similarity methods which incorporate indirect similarity between terms for a text retrieval task. Although their kernels are built on a document-term co-occurrence model, they can also be applied to our candidate-pattern co-occurrence model. The proposed kernel is recursively defined as:

$$\hat{K} = \beta X'\hat{G}X + K, \quad \hat{G} = \beta X\hat{K}X' + G, \quad (7)$$

where $G = XX'$ is the correlation matrix between patterns and β is the factor to ensure that longer range effects decay exponentially. This can be interpreted as augmenting the similarity matrix K through indirect similarities of patterns $\hat{G}$ and vice versa. Semantically related pairs of patterns are expected to be given high correlation in the matrix $\hat{G}$ and this will alleviate the sparseness problem. By

[5] We did not include kanji variants here, because disambiguating kanji readings is a very difficult task, and the majority of the variations in queries are in katakana and Roman alphabet. The framework proposed in this paper, however, can incorporate kanji variants straightforwardly into $\mathrm{ED}(q, c)$ once we have reasonable edit distance functions for kanji variations.

[6] '+' denotes a white space, and '#' indicates where the word of interest is found in a context pattern.

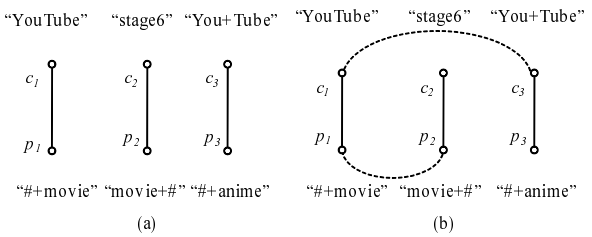


Figure 2: Orthographically Augmented Graph

solving the above recursive definition, one obtains the *von Neumann kernel*:

$$\hat{K}(\beta) = K(I - \beta K)^{-1} = \sum_{t=1}^{\infty} \beta^{t-1} K^t. \quad (8)$$

This can also be interpreted in terms of a random walk in a graph where the nodes correspond to all the candidates and the weight of an edge (i, j) is given by K_{ij}. A simple calculation shows that K_{ij} equals the sum of the products of the edge weights over all possible paths between the nodes corresponding c_i and c_j in the graph. Also, K^t_{ij} corresponds to the probability that a random walk beginning at node c_i ends up at node c_j after t steps, assuming that the entries are all positive and the sum of the connections is 1 at each node. Following this notion, Kandola et al. (2002) proposed another kernel called *exponential kernel*, with alternative faster decay factors:

$$\tilde{K}(\beta) = K \sum_{t=1}^{\infty} \frac{\beta^t K^t}{t!} = K \exp(\beta K). \quad (9)$$

They showed that this alternative kernel achieved a better performance for their text retrieval task. We employed these two kernels to compute distributional similarity for our query correction task.

3.3.3 Orthographically Augmented Kernels

Although semantic relatedness can be partially captured by the semantic kernels introduced in the previous section, they may still have difficulties computing correlations between candidates and patterns especially for only sparsely connected graphs. Take the graph (a) in Fig. 2 for example, which is a simplified yet representative graph topology for candidate-pattern co-occurrence we often encounter. In this case $K = X'X$ equals I, meaning that the connections between candidates and patterns are too sparse to obtain sufficient correlation even when semantic kernels are used.

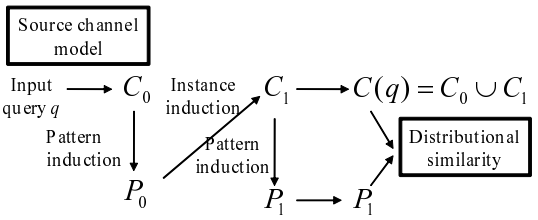


Figure 3: Bootstrapping Additional Candidates

In order to address this issue, we propose to augment the graph by weakly connecting the candidate and pattern nodes as shown in the graph (b) of Fig. 2 based on prior knowledge of orthographic similarity about candidates and patterns. This can be achieved using the following candidate similarity matrix K^+ instead of K:

$$K^+ = \gamma S_C + (1 - \gamma) X' \left[\delta S_P + (1 - \delta) I\right] X \quad (10)$$

where $S_C = \{s_c(i, j)\}$ is the orthographical similarity matrix of candidates in which the (i, j)-element is given by the edit distance based similarity, i.e., $s_c(i, j) = \exp\left[-\mathrm{ED}(c_i, c_j)\right]$. The orthographical similarity matrix of patterns $S_P = \{s_P(i, j)\}$ is defined similarly, i.e., $s_P(i, j) = \exp[-\mathrm{ED}(p_i, p_j)]$. Note that using this similarity matrix K^+ can be interpreted as a random walk process on a bipartite graph as follows. Let C and P as the sets of candidates and patterns. K^+ corresponds to a single walking step from C to C, by either remaining within C with a probability of γ or moving to "the other side" P of the graph with a probability of $1-\gamma$. When the walking remains in C, it is allowed to move to another candidate node following the candidate orthographic similarity S_C. Otherwise it moves to P by the matrix X, chooses either to move within P with a probability γS_P or to stay with a probability $1 - \gamma$, and finally comes back to C by the matrix X'. Multiplication $(K^+)^t$ corresponds to repeating this process t times. Using this similarity, we can define two orthographically augmented semantic kernels which differ in the decaying factors, augmented von Neumann kernel and exponential kernel:

$$\hat{K}^+(\beta) = K^+(I - \beta K^+)^{-1} \quad (11)$$

$$\tilde{K}^+(\beta) = K^+ \exp(\beta K^+). \quad (12)$$

3.4 Bootstrapping Additional Candidates

Now that we have a semantic model, our query correction model can cover query-candidate pairs

which are only semantically related. However, previous work on query correction all used a string distance function and a threshold to restrict the space of potential candidates, allowing only the orthographically similar candidates.

To collect additional candidates, the use of context-based semantic extraction methods would be effective because semantically related candidates are likely to share context with the initial query q, or at least with the initial candidate set C_0. Here we used the *Tchai* algorithm (Komachi and Suzuki, 2008), a modified version of *Espresso* (Pantel and Pennacchiotti, 2006) to collect such candidates. This algorithm starts with initial seed instances, then induces reliable context patterns cooccurring with the seeds, induces instances from the patterns, and iterates this process to obtain categories of semantically related words. Using the candidates in C_0 as the seed instances, one bootstrapping iteration of the *Tchai* algorithm is executed to obtain the semantically related set of instances C_1. The seed instance reliabilities are given by the source channel probabilities $P(c)P(q|c)$. Finally we take the union $C_0 \cup C_1$ to obtain the candidate set $C(q)$. This process is outlined in Fig. 3.

3.5 Maximum Entropy Model

In order to build a unified probabilistic query alteration model, we used the maximum entropy approach of (Beger et al., 1996), which Li et al. (2006) also employed for their query correction task and showed its effectiveness. It defines a conditional probabilistic distribution $P(c|q)$ based on a set of feature functions $f_1, \ldots, f_K$:

$$P(c|q) = \frac{\exp \sum_{i=1}^{K} \lambda_i f_i(c, q)}{\sum_c \exp \sum_{i=1}^{K} \lambda_i f_i(c, q)}, \qquad (13)$$

where $\lambda_1, \ldots, \lambda_K$ are the feature weights. The optimal set of feature weights λ^* can be computed by maximizing the log-likelihood of the training set.

We used the Generalized Iterative Scaling (GIS) algorithm (Darroch and Ratcliff, 1972) to optimize the feature weights. GIS trains conditional probability in Eq. (13), which requires the normalization over all possible candidates. However, the number of all possible candidates C obtained from a query log can be very large, so we only calculated the sum over the candidates in $C(q)$. This is the same approach that Och and Ney (2002) took for statistical machine translation, and Li et al. (2006) for query spelling correction.

We used the following four categories of functions as the features:

1. *Language model feature*, given by the logarithm of the source model probability: $\log P(c)$.
2. *Error model features*, which are composed of three edit distance functions: $-\mathrm{ED}_{\alpha\beta}(q|c)$, $-\mathrm{ED}_{\alpha\beta}(c|q)$, and $-\mathrm{ED}_{\mathrm{hd}}(q, c)$.
3. *Similarity based feature*, computed as the logarithm of distributional similarity between q and c: $\log \mathrm{sim}(q, c)$, which is calcualted using one of the following kernels (Section 3.3): $K, \hat{K}, \tilde{K}, \hat{K}^+$, and $\tilde{K}^+$. The similarity values were normalized to [0, 1] after adding a small discounting factor $\varepsilon = 1.0 \times 10^{-5}$.
4. *Similarity based correction candidate features*, which are binary features with a value of 1 if and only if the frequency of c is higher than that of q, and distributional similarity between them is higher than a certain threshold. Li et al. (2006) used this set of features, and suggested that these features give the evidence that q may be a common misspelling of c. The thresholds on the normalized distributional similarity are enumerated from 0.5 to 0.9 with the interval 0.1.

4 Experiment

4.1 Dataset Creation

For all the experiments conducted in this paper, we used a subset of the Japanese search query logs submitted to Live Search (www.live.com) in November and December of 2007. Queries submitted less than eight times were deleted. The query log we used contained 83,080,257 tokens and 1,038,499 unique queries.

Models of query correction in previous work were trained and evaluated using manually created query-candidate pairs. That is, human annotators were given a set of queries and were asked to provide a correction for each query when it needed to be rewritten. As Cucerzan and Brill (2004) point out, however, this method is seriously flawed in that the intention of the original query is completely lost to the annotator, without which the correction is often impossible: it is not clear if *gogle* should be corrected to *google* or *goggle*, or neither — *gogle* may be a brand new product name. Cucerzan and Brill

therefore performed a second evaluation, where the test data was drawn by sampling the query logs for successive queries (q_1, q_2) by the same user where the edit distance between q_1 and q_2 are within a certain threshold, which are then submitted to annotators for generating the correction. While this method makes the annotation more reliable by relying on user (rather than annotator) reformulation, the task is still overly difficult: going back to the example in Section 1, it is unclear which spelling of 'protein' produces the best search results — it can only be empirically determined. Their method also eliminates all pairs of candidates that are not orthographically similar. We have therefore improved their method in the following manner, making the process more automated and thus more reliable.

We first collected a subset of the query log that contains only those pairs (q_1, q_2) that are issued successively by the same user, q_2 is issued within 3 minutes of q_1, and q_2 resulted in a click of the resulting page while q_1 did not. The last condition adds the evidence that q_2 was a better formulation than q_1. We then ranked the collected query pairs using log-likelihood ratio (LLR) (Dunning, 1993), which measures the dependence between q_1 and q_2 within the context of web queries (Jones et al., 2006b). We randomly sampled 10,000 query pairs with LLR ≥ 200, and submitted them to annotators, who only confirm or reject a query pair as being synonymous. For example, $q_1 =$ *nikon* and $q_2 =$ *canon* are related but not synonymous, while we are reasonably sure $q_1 =$ *ipot* and $q_2 =$ *ipod* are synonymous, given that this pair has a high LLR value. This verification process is extremely fast and consistent across annotators: it takes less than 1 hour to go through 1,000 query pairs, and the inter-annotator agreement rate of two annotators on 2,000 query pairs was 95.7%. We annotated 10,000 query pairs consisting of alpha-numerical and kana characters in this manner. After rejecting non-synonymous pairs and those which do not co-occur with any context patterns, 6,489 pairs remained, and we used 1,243 pairs for testing, 628 as a development set, and 4,618 for training the maximum entropy model.

4.2 Experimental Settings

The performance of query alteration was evaluated based on the following measures (Li et al., 2006). The input queries, correct suggestions, and outputs were matched in a case-insensitive manner.

- *Accuracy*: The number of correct outputs generated by the system divided by the total number of queries in the test set;
- *Recall*: The number of correct suggestions for altered queries divided by the total number of altered queries in the test set;
- *Precision*: The number of correct suggestions for altered queries divided by the total number of alterations made by the system.

The parameters for the kernels, namely, β, γ, and δ, are tuned using the development set. The finally employed values are: $\beta = 0.3$ for $\hat{K}$, $\tilde{K}$, and $\hat{K}^+$, $\beta = 0.2$ for $\tilde{K}^+$, $\gamma = 0.2$ and $\delta = 0.4$ for $\hat{K}^+$, and $\gamma = 0.35$ and $\delta = 0.7$ for $\tilde{K}^+$. In the source channel model, we manually scaled the language probability by a factor of 0.1 to alleviate the bias toward highly frequent candidates.

As the initial candidate set C_0, top-50 instances were selected by the source channel model, and 100 patterns were extracted as P_0 by the *Tchai* iteration after removing generic patterns, which we detected simply by rejecting those which induced more than 200 unique instances. Finally top-30 instances were induced using P_0 to create C_1. Generic instances were not removed in this process because they may still be alterations of input query q. The maximum size of P_1 was set to 2,000, after removing unreliable patterns with reliability smaller than 0.0001.

Table 1: Performance results (%)

Model	Accuracy	Recall	Precision
SC	71.12	39.29	45.09
ME-NoSim	74.58	44.58	52.52
ME-Cos	74.18	**45.84**	50.70
ME-vN	74.34	45.59	52.16
ME-Exp	73.61	44.84	50.57
ME-vN+	75.06	44.33	53.01
ME-Exp+	**75.14**	44.08	**53.52**

4.3 Results

Table 1 shows the evaluation results. SC is the source channel model, while the others are maximum entropy (ME) models with different features. ME-NoSim uses the same features as SC, but considerably outperforms SC in all three measures, confirming the superiority of the ME approach. Decomposing the three edit distance functions into three

separate features in the ME model may also explain the better result. All the ME approaches outperformed SC in accuracy with a statistically significant difference ($p < 0.0001$ on McNemar's test).

The model with the cosine similarity (ME-Cos) in addition to the basic set of features yielded higher recall compared to ME-NoSim, but decreased accuracy and precision, which are more important than recall for our purposes because a false alteration does more damage than no alteration. This is also the case when the kernel-based methods, ME-vN (the von Neumann kernel) and ME-Exp (the exponential kernel), are used in place of the cosine similarity. This shows that using semantic similarity does not always help, which we believe is due to the sparseness of the contextual information used in computing semantic similarity.

On the other hand, ME-vN+ (with augmented von Neumann kernel) and ME-Exp+ (with augmented exponential kernel) increased both accuracy and precision with a slight decrease of recall, compared to the distributional similarity baseline and the non-augmented kernel-based methods. ME-Exp+ was significantly better than ME-Exp ($p < 0.01$).

Note that the accuracy values appear lower than some of the previous results on English (e.g., more than 80% in (Li et al., 2006)), but this is because the dataset creation method we employed tends to over-represent the pairs that lead to alteration: the simplest baseline (= always propose no alteration) performs 67.3% accuracy on our data, in contrast to 83.4% on the data used in (Li et al., 2006).

Manually examining the suggestions made by the system also confirms the effectiveness of our model. For example, the similarity-based models altered the query *ipot* to *ipod*, while the simple ME-NoSim model failed, because it depends too much on the edit distance-based features. We also observed that many of the suggestions made by the system were actually reasonable, even though they were different from the annotated gold standard. For example, ME-vN+ suggests a re-write of the query *2tyann* as 2 ちゃんねる ('2-channel'), while the gold standard was an abbreviated form 2 ちゃん ('2-chan').

To incorporate such possibly correct candidates into account, we conducted a follow-up experiment where we considered multiple reference alterations, created automatically from our data set in the following manner. Suppose that a query q_1 is corrected as q_2, and q_2 is corrected as q_3 in our annotated data. If this is the case, we considered $q_1 \rightarrow q_3$ as a valid alteration as well. By applying this chaining operation up to 5 degrees of separation, we re-created a set of valid alterations for each input query. Note that directionality is important — in the above example, $q_1 \rightarrow q_3$ is valid, while $q_3 \rightarrow q_1$ is not. Table 2 shows the results of evaluation with multiple references. The numbers substantially improved over the single reference cases, as expected, but did not affect the relative performance of each model. Again, the differences in accuracy between the SC and ME models, and ME-Exp and ME-Exp+ were statistically significant ($p < 0.01$).

Table 2: Performance with the multiple reference model

Model	Accuracy	Recall	Precision
SC	75.30	48.61	55.78
ME-NoSim	79.49	56.17	66.17
ME-Cos	79.32	**58.19**	64.35
ME-vN	79.24	57.18	65.42
ME-Exp	78.52	56.42	63.64
ME-vN+	**79.89**	55.67	66.57
ME-Exp+	79.81	54.91	**66.67**

5 Conclusion and future work

In this paper we have presented a unified approach to Japanese query alteration. Our approach draws on previous research in English spelling and query correction, Japanese katakana variation and transliteration, and semantic similarity, and builds a model that makes improvements over previously proposed query correction methods. In particular, the use of orthographically augmented semantic kernels proposed in this paper is general and applicable to other languages, including English, for query alteration, especially when the data sparseness is an issue. In the future, we also plan to investigate other methods, such as PLSI (Hofmann, 1999), to deal with data sparseness in computing semantic similarity.

Acknowledgments

This research was conducted during the first author's internship at Micorosoft Research. We thank the colleagues, especially Dmitriy Belenko, Chris Brockett, Jianfeng Gao, Christian König, and Chris Quirk for their help in conducting this research.

References

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a spelling error model from search query logs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2005)*, pages 955–962.

Eiji Aramaki, Takeshi Imai, Kengo Miyo, and Kazuhiko Ohe. 2008. Orthographic disambiguation incorporating transliterated probability. In *Proceedings in the third International Joint Conference on Natural Language Processing (IJCNLP-2008)*, pages 48–55.

Adam L. Beger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–72.

Slaven Bilac and Hozumi Tanaka. 2004. A hybrid back-transliteration system for japanese. In *Proceedings of the 20th international conference on Computational Linguistics (COLING-2004)*, pages 597–603.

Eric Brill and Robert C. Moore. 2000. An improved error model for noisy channel spelling. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL-2000)*, pages 286–293.

Eric Brill, Gary Kacmarcik, and Chris Brockett. 2001. Automatically harvesting katakana-english term pairs from search engine query logs. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium (NLPRS-2001)*, pages 393–399.

Qing Chen, Mu Li, , and Ming Zhou. 2007. Improving query spelling correction using web search results. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 181–189.

Silviu Cucerzan and Eric Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2004)*, pages 293–300.

Fred Damerau. 1964. A technique for computer detection and correction of spelling errors. *Communication of the ACM*, 7(3):659–664.

J.N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *Annuals of Mathematical Statistics*, 43:1470–1480.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Research and Development in Information Retrieval*, pages 50–57.

Rosie Jones, Kevin Bartz, Pero Subasic, and Benjamin Rey. 2006a. Automatically generating related queries in japanese. *Language Resources and Evaluation (LRE)*, 40(3-4):219–232.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006b. Generating query substitutions. In *Proceedings of the 15th international World Wide Web conference (WWW-06)*, pages 387–396.

Jaz Kandola, John Shawe-Taylor, and Nello Cristianini. 2002. Learning semantic similarity. In *Neural Information Processing Systems (NIPS 15)*, pages 657–664.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24(4):599–612.

Mamoru Komachi and Hisami Suzuki. 2008. Minimally supervised learning of semantic knowledge from query logs. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP-2008)*, pages 358–365.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physice - Doklady*, 10:707–710.

Mu Li, Muhua Zhu, Yang Zhang, and Ming Zhou. 2006. Exploring distributional similarity based models for query spelling correction. In *Proceedings of COLING/ACL-2006*, pages 1025–1032.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING/ACL-1998*, pages 786–774.

Takeshi Masuyama, Satoshi Sekine, and Hiroshi Nakagawa. 2004. Automatic construction of japanese katakana variant list from large corpus. In *Proceedings of Proceedings of the 20th international conference on Computational Linguistics (COLING-2004)*, pages 1214–1219.

Franz Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th annual meeting of ACL*, pages 295–302.

Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pages 1400–1405.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of COLING/ACL-2006*, pages 113–120.

Context-based Message Expansion for Disentanglement of Interleaved Text Conversations

Lidan Wang
Computer Science Dept./UMIACS
University of Maryland, College Park
College Park, MD 20742
lidan@cs.umd.edu

Douglas W. Oard
College of Information Studies/UMIACS
and HLT Center of Excellence
University of Maryland, College Park
College Park, MD 20742
oard@umd.edu

Abstract

Computational processing of text exchanged in interactive venues in which participants engage in simultaneous conversations can benefit from techniques for automatically grouping overlapping sequences of messages into separate conversations, a problem known as "disentanglement." While previous methods exploit both lexical and non-lexical information that exists in conversations for this task, the inter-dependency between the meaning of a message and its temporal and social contexts is largely ignored. Our approach exploits contextual properties (both explicit and hidden) to probabilistically expand each message to provide a more accurate message representation. Extensive experimental evaluations show our approach outperforms the best previously known technique.

1 Introduction

Conversational media such as the text messages found in Internet Relay Chat presents both new opportunities and new challenges. Among the challenges are that individual messages are often quite short, for the reason that conversational participants are able to assemble the required context over the course of a conversation. A natural consequence of this is that many tasks that we would like to perform on conversational media (e.g., search, summarization, or automated response) would benefit from reassembly of individual messages into complete conversations. This task has been studied extensively in the context of email (where it is often referred to as "threading") (Yeh et al., 2006). The extensive metadata associated with email and the relatively rich content of some email messages makes email somewhat of a special case in the broad set of conversation recovery tasks, however. At the opposite extreme, conversation "threading" in multi-party spoken interactions (e.g., meetings) would be a compelling application, but the word error rate of current automated transcription techniques somewhat limits access to the lexical evidence that we know is useful for this task. The recent interest in identifying individual conversations from online-discussions, a task that some refer to as "disentanglement," therefore seems to be something of a middle ground in the research space: computationally tractable, representative to some degree of a broader class of problems, and directly useful as a pre-processing step for a range of important applications.

One way to think of this task is as a clustering problem—we seek to partition the messages into a set of disjoint clusters, where each cluster represents a conversation among a set of participants on a topic. This formulation raises the natural question of how we should design a similarity measure. Since the messages are often too short to be meaningful by themselves, techniques based solely on lexical overlap (e.g., inner products of term vectors weighted by some function of term frequency, document frequency and message length) are unlikely to be successful. For instance, consider the multi-party exchange in Figure 1, in which a single message may not convey much about the topic without considering what has been said before, and who said it.

Fortunately for us, additional sources of evidence

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 200–208,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

(18323 Ricardo) is there a way to emulate input for a program listening on a COM port?
(18911 Azzie) Ricardo: Hello there, how is it going?
(18939 Ricardo) pretty good, just at the office, about to leave. How are you?
(18970 Azzie) well, end of semester work, what could be better?
(18980 Josephina) if it's just reading from /dev/ttyS0 or something you could somehow get it to just read from a named pipe instead
(19034 Ricardo) Josephina: I might just have to end up modifying the entire program...
(19045 Ricardo) so it can read from a different input stream

Figure 1: An example of the text message stream. The number before each author's name denotes the timestamp of the message.

are available. As we describe below, messages are strongly correlated both temporally (i.e., across time) and socially (i.e,, across participants). For example, in our running example in Figure 1, Ricardo's message (19045 Ricardo) "so it can read from a different input stream" elaborates on his previous message (19034 Ricardo) to Josephina. Messages that are close in time and from the same speaker can share related meanings. Similarly, we see that Ricardo's messages to Josephina (19034 Ricardo and 19045 Ricardo) are responses to earlier comments made by Josephina (18980 Josephina), and that fact is signaled by Ricardo invoking Josephena's name. This is an example of social correlation: lexicalized references to identity can also provide useful evidence. If we take social and temporal context into account, we should be able to do better at recognizing conversations than we could using lexical overlap alone.

In recent years, several approaches have been developed for detecting conversational threads in dynamic text streams (Elsner et al., 2008; Shen et al., 2006; Wang et al., 2008). Although they use both lexical and non-lexical information (e.g., time, name mentions in message) for this task, they have ignored the temporal and social contexts a message appears in, which provide valuable cues for interpreting the message. *Correlation clustering* used in a two-step approach (Elsner et al., 2008) exploits message contexts to some degree, but its performance is largely limited by the classifier used in the first-step which computes message similarity without considering the temporal and social contexts of each message.

Our approach exploits contextual properties (both explicit and hidden) to probabilistically expand each message to provide a more accurate message representation. The new representation leads to a much improved performance for conversation disentanglement. We note that this is a general approach and can be applied to the representation of non-chat data that exhibits temporal and social correlations as well. The results that we obtain with this technique are close to the limit of what we can measure using present test collections and evaluation measures. To the best of our knowledge, our work is the first to apply document expansion to the conversation disentanglement problem.

2 Related Work

Previous work in conversation disentanglement (i.e. thread detection) has shown the conventional lexical-based clustering is not suitable for text streams because messages are often too short and incomplete. They focus on using discourse/chat-specific features to bias the lexical-based message similarity (Elsner et al., 2008; Shen et al., 2006; Wang et al., 2008). These features provide the means to link messages that may not have sufficient lexical overlap but are nevertheless likely to be topically related. However, our work is different from them in several aspects:
(1) They treat individual messages as the basic elements for clustering, and ignore the social and temporal contexts of the messages. In our work, each message is probabilistically expanded using reliable information from its contexts and the expanded messages are the basic elements for clustering.
(2) Messages have different amount of explicit information. For example, messages that initiate conversations may have more name mentions than subsequent messages (i.e. for establishing conversations). Previous work only uses what are explicitly present in each message, and clusters may be erroneously assigned for messages that lack enough explicit in-

formation. Our work exploits both explicit and implicit context for each message due to how we define contexts (Section 3.2.1).
(3) Most work imposes a fixed window size for clustering and it may break up long conversations or may not be fine-grained enough for short conversations. Given each message, we use an exponential decay model to naturally encode time effect and assign differential weights to messages in its contexts.

Another thread of related work is document expansion. It was previously studied in (Singhal et al., 1999) in the context of the speech retrieval, helping to overcome limitations in the transcription accuracy by selecting additional terms from lexically similar (text) documents. Document expansion has also been applied to cross-language retrieval in (Levow et al., 2005), in that case to overcome limitations in translation resources. The technique has recently been re-visited (Tao et al., 2006; Kurland et al., 2004; Liu et al., 2004) in the language modeling framework, where lexically related documents are used to enlarge the sample space for a document to improve the accuracy of the estimated document language model. However, these lexical-based approaches are less well suited to conversational interaction, because conversational messages are often short, they therefore may not overlap sufficiently in words with other messages to provide a useful basis for expansion. Our technique can be viewed as an extension of these previous methods to text streams.

Our work is also related to text segmentation (Ji et al., 2003) and meeting segmentation (Malioutov et al., 2006; Malioutov et al., 2007; Galley et al., 2003; Eisenstein et al., 2008). Text segmentation identifies boundaries of topic changes in long text documents, but we form threads of messages from streams consisting of short messages. Meeting conversations are not as highly interleaving as chat conversations, where participants can create a new conversation at any time.

3 Method

This section describes our technique for clustering messages into threads based on the lexical similarity of documents that have been expanded based on social and temporal evidence.

3.1 Context-Free Message Model

To represent the semantic information of messages and threads (clusters of messages), most of the prior approaches build a document representation on each message *alone* (using word features and time-stamp and/or discourse features found in the message). We call such a model a context-free message model. Most commonly, a message is represented as a vector (Salton, 1989). Each dimension corresponds to a separate term. If a term occurs in the message, its value in the vector is non-zero. Several different ways of computing these values, known as term weights, have been developed. One of the best known schemes is tf-idf weighting.

However, in conversational text, a context-free model cannot fully capture the semantics of messages. The meaning of a message is highly dependent on other messages in its context. For example, in our running example in Figure 1, to fully interpret the message 19045 Ricardo, we need to first read his previous message (19034 Ricardo) to Josephina. Further, messages on the same topic may have little or no overlap in words (Figure 1), and the messages between participants are highly interactive and are often too short and incomplete to fully capture a topic on their own.

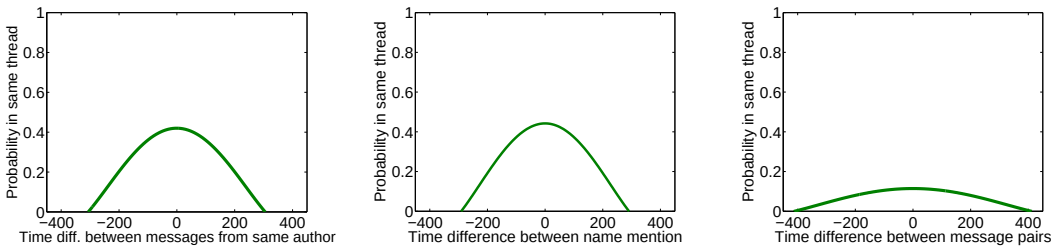
3.2 Context-Sensitive Message Model

Our main idea is to exploit the temporal and social aspects of the conversations to build a context-sensitive document model for each message. We do this by first identifying the temporal and social contexts for each message, then probabilistically expanding the content of each message with selected messages in each context. As we have seen, a message's contexts provide valuable cues for interpreting the message. Finally, we cluster the messages into distinct conversations based on their new representation models.

We present the formal definitions of each context and discuss how to model them in Section 3.2.1. In Section 3.2.2, we show how to efficiently identify the related messages in each context, and how to use them to expand our representation of the message.

3.2.1 Social and Temporal Contexts

Social contexts: we define two kinds of social contexts: author context and conversational context. We



(i) (ii) (iii)

Figure 2: (i) Relationship between messages from the same author (ii) Relationship between messages that mention each other's authors, and (iii) All pairs of messages as a function of time. Estimation is based on training data used in experiments.

explain them in detail below.

Author context: the author context of a message m, denoted by $C_A(m)$, is the set of other messages written by m's author a_m:

$$C_A(m) = \{m_i | a_{m_i} = a_m, m \neq m_i\}$$

Further, because of the nature of human conversations, we would be less surprised to find messages from the same person belonging to the same conversation if they are close in time rather than far apart. This is illustrated in Figure 2(i) [1], which shows the probability that a pair of messages written by the same person belong to the same conversation as a function of the time difference between them. Not surprisingly, messages in m's author context have probabilities which are influenced by their temporal proximity to m.

We use a normal distribution (Figure 2(i)) to encode the notion of author context. Given two messages m_i and m_j written by the *same* author, each with time-stamp t_i and t_j, respectively, the probability that m_j is topically related to m_i given their time difference $d = t_j - t_i$ is:

$$P_a(d) = N(\mu_a, \sigma_a^2) = \frac{1}{\sigma_a\sqrt{2\pi}} e^{-\frac{(d-\mu_a)^2}{2\sigma_a^2}}$$

The exponential decay helps to limit the influence from temporally remote messages. For message m_i, this distribution models the uncertainty that messages in its author context (i.e. other messages m_j from the same author) belong to the same conversation by assigning assigning a high value to m_j if $t_j - t_i$ is small. The mean μ_a is chosen to be zero so that the curve is centered at each message. The variance can be readily estimated from training data.

Conversational context: the second kind of social context is the conversational context, which is constructed from name mentions. As pointed out by previous linguistic studies of discourse, especially analysis of multi-party conversation (ONeill et al., 2003), one key difference between multi-party conversation and typical two-party conversation is the frequency with which participants mention each others' names. Name mentioning is hypothesized as a strategy for participants to compensate for the lack of cues normally present in face-to-face dialogue (ONeill et al., 2003; Elsner et al., 2008). Although infrequent, name mentions (such as Azzie's comments to Ricardo in Figure 1) provide a means for linking two speakers and their messages.

The conversational context of m, $C_C(m)$, is defined to be the set of all messages written by people whose names are mentioned in *any* of a_m's messages (where a_m is the author of m), or who mention a_m in their messages. Let M_a denote all messages written by author a. The conversational context of m is:

$$\begin{aligned} C_C(m) &= \{\forall a\ M_a | mention(a_m, a)\} \\ &\cup \{\forall a\ M_a | mention(a, a_m)\} \end{aligned}$$

where $mention(a_m, a) = true$ if author a_m mentions a in *any* of a_m's messages. $Mention(a, a_m)$ is similarly defined.

Discussion: From the definition, m_j is included in m_i's conversational context if the author of m_i men-

[1] Gaussian kernels shown for illustration purpose in Figure 2 are un-normalized.

tions the author of m_j in any of m_i's messages, or vice versa. For instance, the conversational context for Ricardo's message (19034 Ricardo) in Figure 1 includes the messages from Josephina (18980 Josephina) due to the mentioning of Josephina in his message. However, it may well be the case that m_i does *not* contain any name mentions, e.g. Ricardo's message to Azzie (18939 Ricardo). In this case, if Ricardo is being mentioned by another author (here Azzie asks Ricardo a question by starting with his name in 18939 Azzie), message (18939 Ricardo)'s conversational context will contain *all* of Azzie's messages (18911 and 18970 Azzie) according to the above definition. This intuitively captures the implicit question-answer patterns in conversational speech: Ricardo's subsequent answer is a response to Azzie's comments, hence they are in each other's conversational context.

Our definition also accounts for another source of implicit context. In interactive conversations name mention is a tool for getting people's attention and starting a conversation. Once a participant a_i establishes a conversation with a_j (such that a_i may mention a_j's name in an initial message m_p to a_j), a_i may stop mentioning a_j's name in subsequent messages (m_q) to a_j. This is illustrated in Ricardo's last message to Josephina in Figure 1. Our definition accounts for the *conversation continuity* between a_j and a_i by including messages from a_j in the conversational context of subsequent messages m_q from a_i (note m_q may or may not mention a_j). For instance, message 19045 Ricardo continues the conversation with Josephina from 19034 Ricardo, message 19045 Ricardo thus has Josephina's messages as part of its conversational context.

In general, a person can participate in multiple conversations over time, but as time goes on the topic of interest may shift and the person may start talking to other people. So the messages in the conversational context of m_i due to earlier discussions with other people should be assigned a lower confidence value for m_i. For example, five hours later Ricardo may still be active, but it is unlikely he still chats with Josephina on the same topic, so the earlier messages by Josephina should receive a small confidence value in the conversational context of Ricardo's later messages. We illustrate this idea in Figure 2(ii). It shows the probability that message m_j, where $m_j \in C_C(m_i)$, belongs to the same thread as m_i, given their time difference $t_j - t_i$. This is encoded with a normal probability distribution, $N(\mu_c, \sigma_c)$ where $\mu_c = 0$ and variance is estimated from training data. Let $d = t_j - t_i$, the probability they are topically related given $m_j \in C_C(m_i)$ is:

$$P_c(d) = \frac{1}{\sigma_c\sqrt{2\pi}} e^{-\frac{d^2}{2\sigma_c^2}}$$

Temporal context: temporal context for message m, $C_T(m)$, refers to all other messages:

$$C_T(m) = M \setminus m$$

where M denotes the entire set of messages. The intuition is that nearby messages to m can provide further evidence to the semantics of m. This is illustrated in Figure 2(iii). From the viewpoint of document smoothing, this can also be regarded as using temporally nearby messages to smooth the representation of m. So given m_i, we again model its temporal context by fitting a normal probability distribution $N(\mu_t, \sigma_t)$, so that if $m_j \in C_T(m_i)$ and $d = t_j - t_i$, the probability that m_j is topically related to m_i is:

$$P_t(d) = \frac{1}{\sigma_t\sqrt{2\pi}} e^{-\frac{d^2}{2\sigma_t^2}}$$

3.2.2 Constructing Expanded Messages

We have shown how to use the social and temporal aspects of conversational text to identify and model the contexts of each message, and how to assign confidence values to messages in its contexts. We now show how to use a message's contexts and their associated messages to probabilistically expand the given message. We hypothesize that the expanded message provides a more accurate message representation and that this improved representation can lead to improved accuracy for conversation disentanglement. We will test this hypothesis in the experiment section.

Each message m is represented as a vector of estimated term counts. We expand m using the normalized messages in its contexts. For the expanded message m' of m we estimate the term counts as a linear mixture of term counts from each message in

each context:

$$
\begin{aligned}
c(w, m') &= \alpha c(w, m) + (1 - \alpha)\{ \\
&\quad \lambda_C \sum_{m_j \in C_C(m)} P_c(d_{ji}) \times c(w, m_j) \\
&+ \lambda_A \sum_{m_j \in C_A(m)} P_a(d_{ji}) \times c(w, m_j) \\
&+ \lambda_T \sum_{m_j \in C_T(m)} P_t(d_{ji}) \times c(w, m_j)\}
\end{aligned}
$$

These parameter values are tuned on training data: α controls how much relative weight we give to lexical content of m (0.45 in our experiments), and λ_C, λ_A and λ_T are the relative weights assigned to the conversational, author and temporal contexts (0.6, 0.3, and 0.1 in our experiments, respectively). A context with large variance in its normal density graph should receive a small λ value. This is because a large variance in context k implies more uncertainty on a message m_j being topically related to m while m_j is in the context k of m. In Figure 2, the conversational context (Figure 2(ii)) has the minimum variance among all contexts, hence, it is more accurate for linking messages related in topic and it is assigned a higher λ value (0.6), while the temporal context has the lowest λ value (0.1). Finally, for a message m_j in context k of m_i, $P_k(d_{ji})$ indicates how strongly we believe m_j is topically related to m_i, given their time difference d_{ji}.

Because of the exponential decays of the normal densities that model contexts k, messages in a context will contribute differentially to m_i. Temporally distant messages will have a very low density.

3.3 Single-Pass Clustering

The expanded messages are the basic elements for clustering. The cosine is used to measure similarity:

$$sim(m_i, m_j) = \sum_w \frac{c(w, m_i)c(w, m_j)}{\|m_i\|\|m_j\|}$$

Single-pass clustering is then performed: treat the first message as a single-message cluster T; for each remaining message m compute $\forall T$:

$$sim(m, T) = max_{m_i \in T} sim(m_i, m)$$

For the thread T that maximizes $sim(m, T)$, if $sim(m, T) > t_{sim}$, where t_{sim} is a threshold (0.7 in our experiments) empirically estimated from training data, add m to T; else, start a new cluster containing only m. The time complexity of this algorithm is $O(n^2)$, which is tractable for problems of moderate size.

	Min	**Mean**	**Max**
Number of Conversations	50.00	81.33	128.00
Avg. Conv. Length	6.20	10.60	16.00
Avg. Conv. Density	2.53	2.75	2.92

Table 1: Statistics on the IRC chat transcript data (Elsner et al., 2008). The reported values are based on annotations from six different annotations for the 800 lines of chat transcript.

4 Experiments

The collection used in the experiments consists of real text streams produced in Internet Relay Chat, created by (Elsner et al., 2008) and annotated independently by six annotators. As an upper (human) baseline for each of the three measures reported below, we report the average agreement between all pairs of annotators (i.e., treating one annotator as truth and another as a "system"). For our experiment results, we report the average across all annotators of the agreement between our system and each annotator.

The test collection also contains both a development set and an evaluation set. We used the development set to approximate the normal densities used in our context models and the evaluation set to obtain the results reported below. Some statistics for the 800 annotated messages in the chat transcript of the evaluation collection are shown in Table 1. As that table shows, the average number of active conversation at a given time is 2.75, which makes thread detection a non-trivial task.

4.1 Evaluation Measures

We conduct comparisons using three commonly used evaluation measures for the thread detection task. As a measure of the systems ability to group related messages we report the F-measure (Shen et al., 2006):

$$F = \sum_i \frac{n_i}{n} max_j(F(i, j))$$

where i is a ground-truth conversation with length n_i, and n is the length of entire transcript. $F(i, j)$ is the harmonic mean of recall (fraction of the messages in the i also present in j) and precision (fraction of messages in j also present in i), and F is a weighted sum over all ground-truth conversations (i.e., F is microaveraged).

Two other evaluation measures are "one-to-one accuracy" and "local agreement" (Elsner et al., 2008). "One-to-one accuracy" measures how well we extract whole conversations intact (e.g., as might be required for summarization). It is computed by finding the max-weight bipartite matching between the set of detected threads and the set of real threads, where weight is defined in terms of percentage overlaps for each ground truth and detected thread pair.

Some applications (e.g., real-time monitoring) may not require that we look at entire conversations ar once; in this case a "*local agreement*" measure might make more sense. "loc_3" between system and human annotations as the average (over all possible sets of three consecutive messages) of whether those 3 consecutive messages are assigned consistently by the ground truth and the system. For example, if both the ground truth and the system cluster the first and third messages together and place the second message in a different cluster, then agreement would be recorded.

4.2 Methods Used in Comparison

We compare with the following methods:

Elsner et al. 2008 (best previously known technique): Message similarity is computed with lexical and discourse features, but without document expansion.

Blocks of k: Every consecutive group of k messages is a conversation.

Pause of k: Every pause of k seconds or more separate two conversations.

Speaker: Each speaker's messages are treated as a single conversation.

All different: Each utterance is a separate thread.

All same: The entire transcript is one conversation.

4.3 Results

Figure 3 compares the effectiveness of different schemes in terms of the F measure. We show results

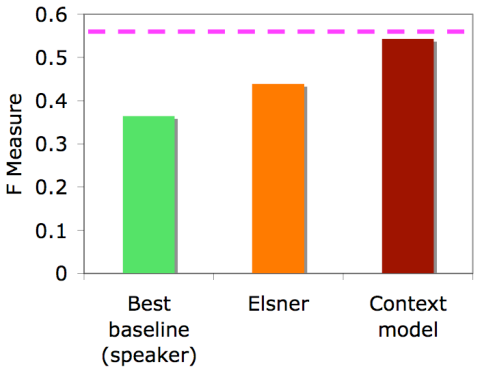


Figure 3: F measure. The dotted line represents inter-annotator agreement.

from the best baseline, Elsner and our technique (which we call the Context model). The average F between human annotators is shown with the dotted line at 0.55; we would expect this to be an upper bound for any model. Our method substantially outperforms the other methods, with a 24% improvement over Elsner and 48% improvement over the best baseline (speaker). Viewed another way, our system achieves 98% of human performance, while Elsner and the best baseline achieve 79% and 66% of that bound, respectively. From this, we can conclude that our Context model is quite effective at clustering messages from same conversation together.

To illustrate the impact of conversation length, we binned the lengths of ground-truth conversations from a single assessor into bins of size 5 (i.e., 3–7 messages, 8–12 messages, . . .; there were no ground truth bins of size 1 or 2). Figure 4 plots the approximated microaveraged F at the center value of each bin (i.e., the F for each ground truth cluster, scaled by the number of messages in the cluster). These fine-grained values provide insight into the contribution of conversations of different sizes to the overall microaveraged F. The Context model performs well for every conversation length, but particularly so for conversations containing 35 or more messages as shown by the widened gap in that region. Long conversations usually have richer social and temporal contexts for each message. The context model can benefit more from drawing evidences from these sources and using them to expand the message, thus makes it possible to group messages of the same

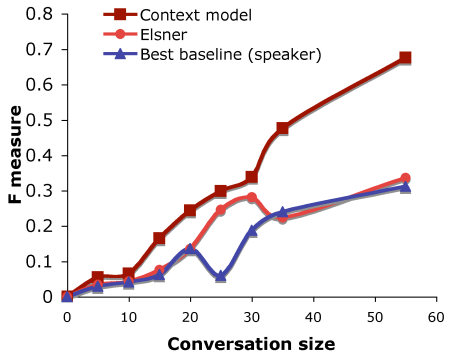


Figure 4: Dependence of F on ground-truth conversation size, in number of messages.

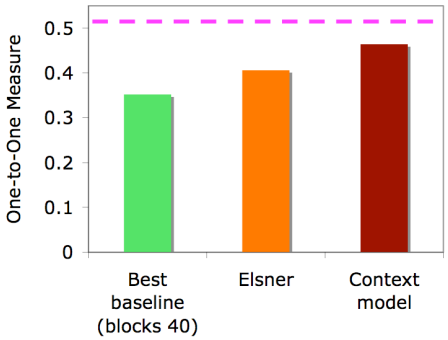


Figure 5: One-to-one measure. The dotted line represents inter-annotator agreement.

conversation together. The other two methods that ignore contextual properties do not do well in comparison.

To measure how well we extract whole conversations intact, Figure 5 shows the results in terms of the one-to-one measure, where each real conversation is matched up with a distinct detected conversation thread. It is computed by max-weight bipartite matching such that the total message overlap is maximized between the sets of detected threads and real threads. The average by this measure between human annotators is 0.53. In this case, the proposed context model achieves an 14% increase over Elsner and 32% increase over the best baseline, and it is within 88% of human performance. This fairly clearly indicates that our Context model can disentangle interleaved conversations relatively well.

Finally, Figure 6 presents the results for "local-3" to evaluate the system's ability to do local annotations. The difference between the best baseline and maximum upper bound is small, implying limited room for potential improvement by any non-baseline techniques. Our result again compares favorably with the previously reported result and the best baseline, although with a smaller margin of 20% over the best baseline and 3% over Elsner as a result of the relatively high baseline for this measure.

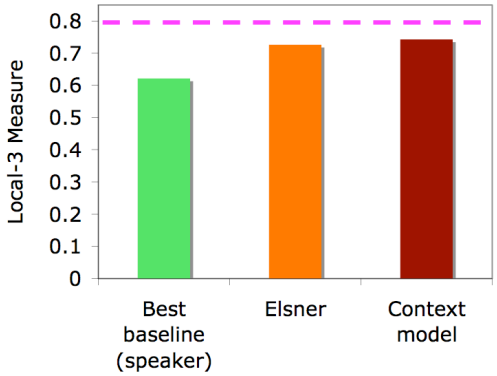


Figure 6: Local-3 measure. The dotted line represents inter-annotator agreement.

5 Conclusion and Future Work

We have presented an approach that exploits contextual properties to probabilistically expand each message to provide a more accurate message representation for dynamic conversations. It is a general approach and can be applied to the representation of non-chat data that exhibits temporal and social correlations as well. For conversation disentanglement, it outperforms the best previously known technique. Our work raises three important questions: (1) to what extent is the single test collection that we have used representative of the broad range of "text chat" applications?, (2) to what extent do the measures we have reported correlate to effective performance of downstream tasks such as summarization or automated response?, and (3) can we re-conceptualize the formalized problem in a way that would result in greater inter-annotator agreement, and hence provide scope for further refinements in our technique. These problems will be the focus of our future work.

References

Micha Elsner and Eugene Charniak. 2008. You talking to me? A Corpus and Algorithm for Conversation Disentanglement. In *ACL 2008: Proceedings of the 46th Annual Meeting on Association for Computational Linguistics*, pages 834-842, Columbus, OH, USA. Association for Computational Linguistics.

Dou Shen, Qiang Yang, Jian-Tao Sun, and Zheng Chen. 2006. Thread Detection in Dynamic Text Message Streams. In *SIGIR 2006: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 35-42, Seattle, WA, USA. Association for Computing Machinery.

Yi-Chia Wang, Mahesh Joshi, William Cohen, and Carolyn Rose. 2008. Recovering Implicit Thread Structure in Newsgroup Style Conversations. In *ICWSM 2008: Proceedings of the 2nd International Conference on Weblogs and Social Media*, pages 152-160, Seattle, WA, USA. Association for the Advancement of Artificial Intelligence.

Tao Tao, Xuanhui Wang, Qiaozhu Mei, and ChengXiang Zhai. 2006. Language Model Information Retrieval with Document Expansion. In *HLT-NAACL 2006: Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, pages 407-414, New York, NY, USA. Association for Computational Linguistics.

Oren Kurland and Lillian Lee. 2004. Corpus Structure, Language Models, and AdHoc Information Retrieval. In *SIGIR 2004: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 194-201, Sheffield, UK. Association for Computing Machinery.

Xiaoyong Liu and W Croft. 2004. Cluster-based Retrieval Using Language Models. In *SIGIR 2004: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186-193, Sheffield, UK. Association for Computing Machinery.

Amit Singhal and Fernando Pereira. 1999. Document Expansion for Speech Retrieval. In *SIGIR 1999: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 34-41, Berkeley, CA, USA. Association for Computing Machinery.

Xiang Ji and Hongyuan Zha 2003. Domain-Independent Text Segmentation using Anisotropic Diffusion and Dynamic Programming. In *SIGIR 2003: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 322-329, Toronto, Canada. Association for Computing Machinery.

Michel Galley, Kathleen McKeown, Eric Lussier, and Hongyan Jing. 2003. Discourse Segmentation of Multi-Party Conversation. In *ACL 2003: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 562-569, Sapporo, Japan. Association for Computational Linguistics.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian Unsupervised Topic Segmentation. In *EMNLP 2008: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 334-343, Honolulu, Hawaii, USA. Association for Computational Linguistics.

Igor Malioutov and Regina Barzilay 2006. Minimum-Cut Model for Spoken Lecture Segmentation. In *ACL 2006: Proceedings of the 44rd Annual Meeting of the Association for Computational Linguistics*, pages 25-32, Sydney, Australia. Association for Computational Linguistics.

Igor Malioutov, Alex Park, Regina Barzilay, and James Glass. 2007. Making Sense of Sound: Unsupervised Topic Segmentation over Acoustic Input. In *ACL 2007: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 504-511, Prague, Czech Republic. Association for Computational Linguistics.

Jen-Yuan Yeh and Aaron Harnly. 2006. Email Thread Reassembly Using Similarity Matching. In *CEAS 2006: The 3rd Conference on Email and Anti-Spam*, pages 64-71, Mountain View, CA, USA.

Jacki ONeill and David Martin. 2003. Text Chat in Action. In *ACM SIGGROUP 2003: Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*, pages 40-49, New York, NY, USA. ACM Press.

Gerard Salton. 1989. Automatic Text Processing: the Transformation, Analysis and Retrieval of Information by Computer. *Addison-Wesley Longman Publishing Co., Inc.*, Boston, MA, USA, 1989.

Gina-Anne Levow, Douglas Oard, and Philip Resnik. 2005. Dictionary-based techniques for cross-language information retrieval. In *Information Processing and Management Special Issue: Cross-Language Information Retrieval*, 41(3): 523-547.

Unsupervised Morphological Segmentation with Log-Linear Models

Hoifung Poon[*]
Dept. of Computer Sci. & Eng.
University of Washington
Seattle, WA 98195
hoifung@cs.washington.edu

Colin Cherry
Microsoft Research
Redmond, WA 98052
colinc@microsoft.com

Kristina Toutanova
Microsoft Research
Redmond, WA 98052
kristout@microsoft.com

Abstract

Morphological segmentation breaks words into morphemes (the basic semantic units). It is a key component for natural language processing systems. Unsupervised morphological segmentation is attractive, because in every language there are virtually unlimited supplies of text, but very few labeled resources. However, most existing model-based systems for unsupervised morphological segmentation use directed generative models, making it difficult to leverage arbitrary overlapping features that are potentially helpful to learning. In this paper, we present the first log-linear model for unsupervised morphological segmentation. Our model uses overlapping features such as morphemes and their contexts, and incorporates exponential priors inspired by the minimum description length (MDL) principle. We present efficient algorithms for learning and inference by combining contrastive estimation with sampling. Our system, based on monolingual features only, outperforms a state-of-the-art system by a large margin, even when the latter uses bilingual information such as phrasal alignment and phonetic correspondence. On the Arabic Penn Treebank, our system reduces F1 error by 11% compared to Morfessor.

1 Introduction

The goal of morphological segmentation is to segment words into *morphemes*, the basic syntactic/semantic units. This is a key subtask in many NLP applications, including machine translation, speech recognition and question answering. Past approaches include rule-based morphological analyzers (Buckwalter, 2004) and supervised learning (Habash and Rambow, 2005). While successful, these require deep language expertise and a long and laborious process in system building or labeling.

Unsupervised approaches are attractive due to the the availability of large quantities of unlabeled text, and unsupervised morphological segmentation has been extensively studied for a number of languages (Brent et al., 1995; Goldsmith, 2001; Dasgupta and Ng, 2007; Creutz and Lagus, 2007). The lack of supervised labels makes it even more important to leverage rich features and global dependencies. However, existing systems use directed generative models (Creutz and Lagus, 2007; Snyder and Barzilay, 2008b), making it difficult to extend them with arbitrary overlapping dependencies that are potentially helpful to segmentation.

In this paper, we present the first log-linear model for unsupervised morphological segmentation. Our model incorporates simple priors inspired by the minimum description length (MDL) principle, as well as overlapping features such as morphemes and their contexts (e.g., in Arabic, the string *Al* is likely a morpheme, as is any string between *Al* and a word boundary). We develop efficient learning and inference algorithms using a novel combination of two ideas from previous work on unsupervised learning with log-linear models: contrastive estimation (Smith and Eisner, 2005) and sampling (Poon and Domingos, 2008).

We focus on inflectional morphology and test our

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

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 209–217,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

approach on datasets in Arabic and Hebrew. Our system, using monolingual features only, outperforms Snyder & Barzilay (2008b) by a large margin, even when their system uses bilingual information such as phrasal alignment and phonetic correspondence. On the Arabic Penn Treebank, our system reduces F1 error by 11% compared to Morfessor Categories-MAP (Creutz and Lagus, 2007). Our system can be readily applied to supervised and semi-supervised learning. Using a fraction of the labeled data, it already outperforms Snyder & Barzilay's supervised results (2008a), which further demonstrates the benefit of using a log-linear model.

2 Related Work

There is a large body of work on the unsupervised learning of morphology. In addition to morphological segmentation, there has been work on unsupervised morpheme analysis, where one needs to determine features of word forms (Kurimo et al., 2007) or identify words with the same lemma by modeling stem changes (Schone and Jurafsky, 2001; Goldsmith, 2001). However, we focus our review specifically on morphological segmentation.

In the absence of labels, unsupervised learning must incorporate a strong learning bias that reflects prior knowledge about the task. In morphological segmentation, an often-used bias is the minimum description length (MDL) principle, which favors compact representations of the lexicon and corpus (Brent et al., 1995; Goldsmith, 2001; Creutz and Lagus, 2007). Other approaches use statistics on morpheme context, such as conditional entropy between adjacent n-grams, to identify morpheme candidates (Harris, 1955; Keshava and Pitler, 2006). In this paper, we incorporate both intuitions into a simple yet powerful model, and show that each contributes significantly to performance.

Unsupervised morphological segmentation systems also differ from the engineering perspective. Some adopt a pipeline approach (Schone and Jurafsky, 2001; Dasgupta and Ng, 2007; Demberg, 2007), which works by first extracting candidate affixes and stems, and then segmenting the words based on the candidates. Others model segmentation using a joint probabilistic distribution (Goldwater et al., 2006; Creutz and Lagus, 2007; Snyder and Barzilay, 2008b); they learn the model parameters from unlabeled data and produce the most probable segmentation as the final output. The latter approach is arguably more appealing from the modeling standpoint and avoids error propagation along the pipeline. However, most existing systems use directed generative models; Creutz & Lagus (2007) used an HMM, while Goldwater et al. (2006) and Snyder & Barzilay (2008b) used Bayesian models based on Pitman-Yor or Dirichlet processes. These models are difficult to extend with arbitrary overlapping features that can help improve accuracy.

In this work we incorporate novel overlapping contextual features and show that they greatly improve performance. Non-overlapping contextual features previously have been used in directed generative models (in the form of Markov models) for unsupervised morphological segmentation (Creutz and Lagus, 2007) or word segmentation (Goldwater et al., 2007). In terms of feature sets, our model is most closely related to the constituent-context model proposed by Klein and Manning (2001) for grammar induction. If we exclude the priors, our model can also be seen as a semi-Markov conditional random field (CRF) model (Sarawagi and Cohen, 2004). Semi-Markov CRFs previously have been used for supervised word segmentation (Andrew, 2006), but not for unsupervised morphological segmentation.

Unsupervised learning with log-linear models has received little attention in the past. Two notable exceptions are Smith & Eisner (2005) for POS tagging, and Poon & Domingos (2008) for coreference resolution. Learning with log-linear models requires computing the normalization constant (a.k.a. the partition function) Z. This is already challenging in supervised learning. In unsupervised learning, the difficulty is further compounded by the absence of supervised labels. Smith & Eisner (2005) proposed *contrastive estimation*, which uses a small neighborhood to compute Z. The neighborhood is carefully designed so that it not only makes computation easier but also offers sufficient contrastive information to aid unsupervised learning. Poon & Domingos (2008), on the other hand, used sampling to approximate Z.[1] In this work, we benefit from both techniques: contrastive estimation creates a manageable,

[1] Rosenfeld (1997) also did this for language modeling.

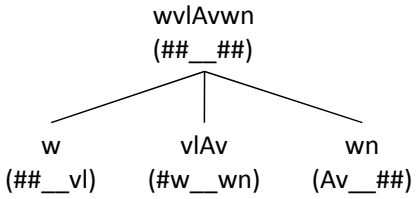


Figure 1: The morpheme and context (in parentheses) features for the segmented word *w-vlAv-wn*.

informative Z, while sampling enables the use of powerful global features.

3 Log-Linear Model for Unsupervised Morphological Segmentation

Central to our approach is a log-linear model that defines the joint probability distribution for a corpus (i.e., the words) and a segmentation on the corpus. The core of this model is a morpheme-context model, with one feature for each morpheme,[2] and one feature for each morpheme context. We represent contexts using the n-grams before and after the morpheme, for some constant n. To illustrate this, a segmented Arabic corpus is shown below along with its features, assuming we are tracking bigram contexts. The segmentation is indicated with hyphens, while the hash symbol (#) represents the word boundary.

Segmented Corpus hnAk w-vlAv-wn bn-w Al-ywm Al-jmAEp

Morpheme Feature:Value hnAk:1 w:2 vlAv:1 wn:1 bn:1 Al:2 ywm:1 jmAEp:1 hnAk:1 wvlAvwn:1 bnw:1 Alywm:1 Aljmaep:1

Bigram Context Feature:Value ##_vl:1 #w_wn:1 Av_##:1 ##_w#:1 bn_##:1 ##_yw:1 Al_##:2 ##_jm:1 ##_##:5

Furthermore, the corresponding features for the segmented word *w-vlAv-wn* are shown in Figure 1.

Each feature is associated with a weight, which correlates with the likelihood that the corresponding morpheme or context marks a valid morphological segment. Such overlapping features allow us to capture rich segmentation regularities. For example, given the Arabic word *Alywm*, to derive its correct segmentation *Al-ywm*, it helps to know that *Al* and *ywm* are likely morphemes whereas *Aly* or *lyw* are not; it also helps to know that *Al_##* or *##_yw* are likely morpheme contexts whereas *ly_##* or *##_wm* are not. Ablation tests verify the importance of these overlapping features (see Section 7.2).

Our morpheme-context model is inspired by the constituent-context model (CCM) proposed by Klein and Manning (2001) for grammar induction. The morphological segmentation of a word can be viewed as a flat tree, where the root node corresponds to the word and the leaves correspond to morphemes (see Figure 1). The CCM uses unigrams for context features. For this task, however, we found that bigrams and trigrams lead to much better accuracy. We use trigrams in our full model.

For learning, one can either view the corpus as a collection of word *types* (unique words) or *tokens* (word occurrences). Some systems (e.g., Morfessor) use token frequency for parameter estimation. Our system, however, performs much better using word types. This has also been observed for other morphological learners (Goldwater et al., 2006). Thus we use types in learning and inference, and effectively enforce the constraint that words can have only one segmentation per type. Evaluation is still based on tokens to reflect the performance in real applications.

In addition to the features of the morpheme-context model, we incorporate two priors which capture additional intuitions about morphological segmentations. First, we observe that the number of distinct morphemes used to segment a corpus should be small. This is achieved when the same morphemes are re-used across many different words. Our model incorporates this intuition by imposing a **lexicon prior**: an exponential prior with negative weight on the length of the morpheme lexicon. We define the lexicon to be the set of unique morphemes identified by a complete segmentation of the corpus, and the lexicon length to be the total number of characters in the lexicon. In this way, we can simultaneously emphasize that a lexicon should contain few unique morphemes, and that those morphemes should be short. However, the lexicon prior alone incorrectly favors the trivial segmentation that shatters each word into characters, which results in the smallest lexicon possible (single characters). Therefore, we also impose a **corpus prior**: an exponential prior on the number of mor-

[2]The word as a whole is also treated as a morpheme in itself.

phemes used to segment each word in the corpus, which penalizes over-segmentation. We notice that longer words tend to have more morphemes. Therefore, each word's contribution to this prior is normalized by the word's length in characters (e.g., the segmented word *w-vlAv-wn* contributes 3/7 to the total corpus size). Notice that it is straightforward to incorporate such a prior in a log-linear model, but much more challenging to do so in a directed generative model. These two priors are inspired by the minimum description length (MDL) length principle; the lexicon prior favors fewer morpheme types, whereas the corpus prior favors fewer morpheme tokens. They are vital to the success of our model, providing it with the initial inductive bias.

We also notice that often a word is decomposed into a stem and some prefixes and suffixes. This is particularly true for languages with predominantly inflectional morphology, such as Arabic, Hebrew, and English. Thus our model uses separate lexicons for prefixes, stems, and suffixes. This results in a small but non-negligible accuracy gain in our experiments. We require that a stem contain at least two characters and no fewer characters than any affixes in the same word.[3] In a given word, when a morpheme is identified as the stem, any preceding morpheme is identified as a prefix, whereas any following morpheme as a suffix. The sample segmented corpus mentioned earlier induces the following lexicons:

Prefix w Al
Stem hnAk vlAv bn ywm jmAEp
Suffix wn w

Before presenting our formal model, we first introduce some notation. Let W be a corpus (i.e., a set of words), and S be a segmentation that breaks each word in W into prefixes, a stem, and suffixes. Let σ be a string (character sequence). Each occurrence of σ will be in the form of $\psi_1\sigma\psi_2$, where ψ_1, ψ_2 are the adjacent character n-grams, and $c = (\psi_1, \psi_2)$ is the context of σ in this occurrence. Thus a segmentation can be viewed as a set of morpheme strings and their contexts. For a string x, $L(x)$ denotes the number of characters in x; for a word w, $M_S(w)$ denotes the number of morphemes in w given the segmentation S; $Pref(W,S), Stem(W,S), Suff(W,S)$ denote the lexicons of prefixes, stems, and suffixes induced by S for W. Then, our model defines a joint probability distribution over a restricted set of W and S:

$$P_\theta(W,S) \;=\; \frac{1}{Z}\cdot u_\theta(W,S)$$

where

$$\begin{aligned} u_\theta(W,S) \;=\; \exp\Big(\; & \sum_\sigma \lambda_\sigma f_\sigma(S) + \sum_c \lambda_c f_c(S) \\ & + \alpha \cdot \sum_{\sigma\in Pref(W,S)} L(\sigma) \\ & + \alpha \cdot \sum_{\sigma\in Stem(W,S)} L(\sigma) \\ & + \alpha \cdot \sum_{\sigma\in Suff(W,S)} L(\sigma) \\ & + \beta \cdot \sum_{w\in W} M_S(w)/L(w) \;\Big) \end{aligned}$$

Here, $f_\sigma(S)$ and $f_c(S)$ are respectively the occurrence counts of morphemes and contexts under S, and $\theta = (\lambda_\sigma, \lambda_c : \sigma, c)$ are their feature weights. α, β are the weights for the priors. Z is the normalization constant, which sums over a set of corpora and segmentations. In the next section, we will define this set for our model and show how to efficiently perform learning and inference.

4 Unsupervised Learning

As mentioned in Smith & Eisner (2005), learning with probabilistic models can be viewed as moving probability mass to the observed data. The question is from where to take this mass. For log-linear models, the answer amounts to defining the set that Z sums over. We use contrastive estimation and define the set to be a neighborhood of the observed data. The instances in the neighborhood can be viewed as pseudo-negative examples, and learning seeks to discriminate them from the observed instances.

Formally, let W^* be the observed corpus, and let $N(\cdot)$ be a function that maps a string to a set of strings; let $N(W^*)$ denote the set of all corpora that can be derived from W^* by replacing every word $w \in W^*$ with one in $N(w)$. Then,

$$Z = \sum_{W\in N(W^*)} \sum_S u(W,S).$$

[3] In a segmentation where several morphemes have the maximum length, any of them can be identified as the stem, each resulting in a distinct segmentation.

Unsupervised learning maximizes the log-likelihood of observing W^*

$$L_\theta(W^*) = \log \sum_S P(W^*, S)$$

We use gradient descent for this optimization; the partial derivatives for feature weights are

$$\frac{\partial}{\partial \lambda_i} L_\theta(W^*) = E_{S|W^*}[f_i] - E_{S,W}[f_i]$$

where i is either a string σ or a context c. The first expected count ranges over all possible segmentations while the words are fixed to those observed in W^*. For the second expected count, the words also range over the neighborhood.

Smith & Eisner (2005) considered various neighborhoods for unsupervised POS tagging, and showed that the best neighborhoods are `TRANS1` (transposing any pair of adjacent words) and `DELORTRANS1` (deleting any word or transposing any pair of adjacent words). We can obtain their counterparts for morphological segmentation by simply replacing "words" with "characters". As mentioned earlier, the instances in the neighborhood serve as pseudo-negative examples from which probability mass can be taken away. In this regard, `DELORTRANS1` is suitable for POS tagging since deleting a word often results in an ungrammatical sentence. However, in morphology, a word less a character is often a legitimate word too. For example, deleting *l* from the Hebrew word *lyhwh* (to the lord) results in *yhwh* (the lord). Thus `DELORTRANS1` forces legal words to compete against each other for probability mass, which seems like a misguided objective. Therefore, in our model we use `TRANS1`. It is suited for our task because transposing a pair of adjacent characters usually results in a non-word.

To combat overfitting in learning, we impose a Gaussian prior (L_2 regularization) on all weights.

5 Supervised Learning

Our learning algorithm can be readily applied to supervised or semi-supervised learning. Suppose that gold segmentation is available for some words, denoted as S^*. If S^* contains gold segmentations for all words in W, we are doing supervised learning; otherwise, learning is semi-supervised. Training now maximizes $L_\theta(W^*, S^*)$; the partial derivatives become

$$\frac{\partial}{\partial \lambda_i} L_\theta(W^*, S^*) = E_{S|W^*,S^*}[f_i] - E_{S,W}[f_i]$$

The only difference in comparison with unsupervised learning is that we fix the known segmentation when computing the first expected counts. In Section 7.3, we show that when labels are available, our model also learns much more effectively than a directed graphical model.

6 Inference

In Smith & Eisner (2005), the objects (sentences) are independent from each other, and exact inference is tractable. In our model, however, the lexicon prior renders all objects (words) interdependent in terms of segmentation decisions. Consider the simple corpus with just two words: *Alrb*, *lAlrb*. If *lAlrb* is segmented into *l-Al-rb*, *Alrb* can be segmented into *Al-rb* without paying the penalty imposed by the lexicon prior. If, however, *lAlrb* remains a single morpheme, and we still segment *Alrb* into *Al-rb*, then we introduce two new morphemes into the lexicons, and we will be penalized by the lexicon prior accordingly. As a result, we must segment the whole corpus jointly, making exact inference intractable. Therefore, we resort to approximate inference. To compute $E_{S|W^*}[f_i]$, we use Gibbs sampling. To derive a sample, the procedure goes through each word and samples the next segmentation conditioned on the segmentation of all other words. With m samples $S_1, \cdots, S_m$, the expected count can be approximated as

$$E_{S|W^*}[f_i] \approx \frac{1}{m} \sum_j f_i(S_j)$$

There are 2^{n-1} ways to segment a word of n characters. To sample a new segmentation for a particular word, we need to compute conditional probability for each of these segmentations. We currently do this by explicit enumeration.[4] When n is large,

[4]These segmentations could be enumerated implicitly using the dynamic programming framework employed by semi-Markov CRFs (Sarawagi and Cohen, 2004). However, in such a setting, our lexicon prior would likely need to be approximated. We intend to investigate this in future work.

this is very expensive. However, we observe that the maximum number of morphemes that a word contains is usually a small constant for many languages; in the Arabic Penn Treebank, the longest word contains 14 characters, but the maximum number of morphemes in a word is only 5. Therefore, we impose the constraint that a word can be segmented into no more than k morphemes, where k is a language-specific constant. We can determine k from prior knowledge or use a development set. This constraint substantially reduces the number of segmentation candidates to consider; with $k = 5$, it reduces the number of segmentations to consider by almost 90% for a word of 14 characters.

$E_{S,W}[f_i]$ can be computed by Gibbs sampling in the same way, except that in each step we also sample the next word from the neighborhood, in addition to the next segmentation.

To compute the most probable segmentation, we use deterministic annealing. It works just like a sampling algorithm except that the weights are divided by a *temperature*, which starts with a large value and gradually drops to a value close to zero. To make burn-in faster, when computing the expected counts, we initialize the sampler with the most probable segmentation output by annealing.

7 Experiments

We evaluated our system on two datasets. Our main evaluation is on a multi-lingual dataset constructed by Snyder & Barzilay (2008a; 2008b). It consists of 6192 short parallel phrases in Hebrew, Arabic, Aramaic (a dialect of Arabic), and English. The parallel phrases were extracted from the Hebrew Bible and its translations via word alignment and post-processing. For Arabic, the gold segmentation was obtained using a highly accurate Arabic morphological analyzer (Habash and Rambow, 2005); for Hebrew, from a Bible edition distributed by Westminster Hebrew Institute (Groves and Lowery, 2006). There is no gold segmentation for English and Aramaic. Like Snyder & Barzilay, we evaluate on the Arabic and Hebrew portions only; unlike their approach, our system does not use any bilingual information. We refer to this dataset as **S&B** . We also report our results on the Arabic Penn Treebank (**ATB**), which provides gold segmentations for an Arabic corpus with about 120,000 Arabic words.

As in previous work, we report recall, precision, and F1 over segmentation points. We used 500 phrases from the S&B dataset for feature development, and also tuned our model hyperparameters there. The weights for the lexicon and corpus priors were set to $\alpha = -1$, $\beta = -20$. The feature weights were initialized to zero and were penalized by a Gaussian prior with $\sigma^2 = 100$. The learning rate was set to 0.02 for all experiments, except the full Arabic Penn Treebank, for which it was set to 0.005.[5] We used 30 iterations for learning. In each iteration, 200 samples were collected to compute each of the two expected counts. The sampler was initialized by running annealing for 2000 samples, with the temperature dropping from 10 to 0.1 at 0.1 decrements. The most probable segmentation was obtained by running annealing for 10000 samples, using the same temperature schedule. We restricted the segmentation candidates to those with no greater than five segments in all experiments.

7.1 Unsupervised Segmentation on S&B

We followed the experimental set-up of Snyder & Barzilay (2008b) to enable a direct comparison. The dataset is split into a training set with 4/5 of the phrases, and a test set with the remaining 1/5. First, we carried out unsupervised learning on the training data, and computed the most probable segmentation for it. Then we fixed the learned weights and the segmentation for training, and computed the most probable segmentation for the test set, on which we evaluated.[6] Snyder & Barzilay (2008b) compared several versions of their systems, differing in how much bilingual information was used. Using monolingual information only, their system (S&B-MONO) trails the state-of-the-art system Morfessor; however, their best system (S&B-BEST), which uses bilingual information that includes phrasal alignment and phonetic correspondence between Arabic and Hebrew, outperforms Morfessor and achieves the state-of-the-art results on this dataset.

[5] The ATB set is more than an order of magnitude larger and requires a smaller rate.

[6] With unsupervised learning, we can use the entire dataset for training since no labels are provided. However, this set-up is necessary for S&B's system because they used bilingual information in training, which is not available at test time.

ARABIC	Prec.	Rec.	F1
S&B-MONO	53.0	78.5	63.2
S&B-BEST	67.8	77.3	72.2
FULL	76.0	80.2	**78.1**
HEBREW	Prec.	Rec.	F1
S&B-MONO	55.8	64.4	59.8
S&B-BEST	64.9	62.9	63.9
FULL	67.6	66.1	**66.9**

Table 1: Comparison of segmentation results on the S&B dataset.

Table 1 compares our system with theirs. Our system outperforms both S&B-MONO and S&B-BEST by a large margin. For example, on Arabic, our system reduces F1 error by 21% compared to S&B-BEST, and by 40% compared to S&B-MONO. This suggests that the use of monolingual morpheme context, enabled by our log-linear model, is more helpful than their bilingual cues.

7.2 Ablation Tests

To evaluate the contributions of the major components in our model, we conducted seven ablation tests on the S&B dataset, each using a model that differed from our full model in one aspect. The first three tests evaluate the effect of priors, whereas the next three test the effect of context features. The last evaluates the impact of using separate lexicons for affixes and stems.

NO-PRIOR The priors are not used.
NO-COR-PR The corpus prior is not used.
NO-LEX-PR The lexicon prior is not used.
NO-CONTEXT Context features are not used.
UNIGRAM Unigrams are used in context.
BIGRAM Bigrams are used in context.
SG-LEXICON A single lexicon is used, rather than three distinct ones for the affixes and stems.

Table 2 presents the ablation results in comparison with the results of the full model. When some or all priors are excluded, the F1 score drops substantially (over 10 points in all cases, and over 40 points in some). In particular, excluding the corpus prior, as in NO-PRIOR and NO-COR-PR, results in over-segmentation, as is evident from the high recalls and low precisions. When the corpus prior is enacted but not the lexicon priors (NO-LEX-PR), precision is much higher, but recall is low; the system now errs on under-segmentation because recurring strings are often not identified as morphemes.

ARABIC	Prec.	Rec.	F1
FULL	76.0	80.2	**78.1**
NO-PRIOR	24.6	89.3	38.6
NO-COR-PR	23.7	87.4	37.2
NO-LEX-PR	79.1	51.3	62.3
NO-CONTEXT	71.2	62.1	66.3
UNIGRAM	71.3	76.5	73.8
BIGRAM	73.1	78.4	75.7
SG-LEXICON	72.8	82.0	77.1
HEBREW	Prec.	Rec.	F1
FULL	67.6	66.1	66.9
NO-PRIOR	34.0	89.9	49.4
NO-COR-PR	35.6	90.6	51.1
NO-LEX-PR	65.9	49.2	56.4
NO-CONTEXT	63.0	47.6	54.3
UNIGRAM	63.0	63.7	63.3
BIGRAM	69.5	66.1	**67.8**
SG-LEXICON	67.4	65.7	66.6

Table 2: Ablation test results on the S&B dataset.

A large accuracy drop (over 10 points in F1 score) also occurs when the context features are excluded (NO-CONTEXT), which underscores the importance of these overlapping features. We also notice that the NO-CONTEXT model is comparable to the S&B-MONO model; they use the same feature types, but different priors. The accuracies of the two systems are comparable, which suggests that we did not sacrifice accuracy by trading the more complex and restrictive Dirichlet process prior for exponential priors. A priori, it is unclear whether using contexts larger than unigrams would help. While potentially beneficial, they also risk aggravating the data sparsity and making our model more prone to overfitting. For this problem, however, enlarging the context (using higher n-grams up to trigrams) helps substantially. For Arabic, the highest accuracy is attained by using trigrams, which reduces F1 error by 16% compared to unigrams; for Hebrew, by using bigrams, which reduces F1 error by 17%. Finally, it helps to use separate lexicons for affixes and stems, although the difference is small.

ARABIC	%Lbl.	Prec.	Rec.	F1
S&B-MONO-S	100	73.2	92.4	81.7
S&B-BEST-S	200	77.8	92.3	84.4
FULL-S	25	84.9	85.5	85.2
	50	88.2	86.8	87.5
	75	89.6	86.4	87.9
	100	91.7	88.5	**90.0**
HEBREW	%Lbl.	Prec.	Rec.	F1
S&B-MONO-S	100	71.4	79.1	75.1
S&B-BEST-S	200	76.8	79.2	78.0
FULL-S	25	78.7	73.3	75.9
	50	82.8	74.6	78.4
	75	83.1	77.3	80.1
	100	83.0	78.9	**80.9**

Table 3: Comparison of segmentation results with supervised and semi-supervised learning on the S&B dataset.

7.3 Supervised and Semi-Supervised Learning

To evaluate our system in the supervised and semi-supervised learning settings, we report the performance when various amounts of labeled data are made available during learning, and compare them to the results of Snyder & Barzilay (2008a). They reported results for supervised learning using monolingual features only (S&B-MONO-S), and for supervised bilingual learning with labels for both languages (S&B-BEST-S). On both languages, our system substantially outperforms both S&B-MONO-S and S&B-BEST-S. E.g., on Arabic, our system reduces F1 errors by 46% compared to S&B-MONO-S, and by 36% compared to S&B-BEST-S. Moreover, with only one-fourth of the labeled data, our system already outperforms S&B-MONO-S. This demonstrates that our log-linear model is better suited to take advantage of supervised labels.

7.4 Arabic Penn Treebank

We also evaluated our system on the Arabic Penn Treebank (ATB). As is common in unsupervised learning, we trained and evaluated on the entire set. We compare our system with Morfessor (Creutz and Lagus, 2007).[7] In addition, we compare with Morfessor Categories-MAP, which builds on Morfessor and conducts an additional greedy search specifically tailored to segmentation. We found that it performs much better than Morfessor on Arabic but worse on Hebrew. To test each system in a low-data setting, we also ran experiments on the set containing the first 7,000 words in ATB with at least two characters (ATB-7000). Table 4 shows the results. Morfessor performs rather poorly on ATB-7000. Morfessor Categories-MAP does much better, but its performance is dwarfed by our system, which further cuts F1 error by half. On the full ATB dataset, Morfessor performs even worse, whereas Morfessor Categories-MAP benefits from the larger dataset and achieves an F1 of 74.9. Still, our system substantially outperforms it, further reducing F1 error by 11%.[8]

ATB-7000	Prec.	Rec.	F1
MORFESSOR-1.0	70.6	34.3	46.1
MORFESSOR-MAP	86.9	46.4	60.5
FULL	83.4	77.3	**80.2**
ATB	Prec.	Rec.	F1
MORFESSOR-1.0	80.7	20.4	32.6
MORFESSOR-MAP	77.4	72.6	74.9
FULL	88.5	69.2	**77.7**

Table 4: Comparison of segmentation results on the Arabic Penn Treebank.

8 Conclusion

This paper introduces the first log-linear model for unsupervised morphological segmentation. It leverages overlapping features such as morphemes and their contexts, and enables easy extension to incorporate additional features and linguistic knowledge. For Arabic and Hebrew, it outperforms the state-of-the-art systems by a large margin. It can also be readily applied to supervised or semi-supervised learning when labeled data is available. Future directions include applying our model to other inflectional and agglutinative languages, modeling internal variations of morphemes, leveraging parallel data in multiple languages, and combining morphological segmentation with other NLP tasks, such as machine translation.

[7] We cannot compare with Snyder & Barzilay's system as its strongest results require bilingual data, which is not available.

[8] Note that the ATB and ATB-7000 experiments each measure accuracy on their entire training set. This difference in testing conditions explains why some full ATB results are lower than ATB-7000.

References

Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Michael R. Brent, Sreerama K. Murthy, and Andrew Lundberg. 1995. Discovering morphemic suffixes: A case study in minimum description length induction. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*.

Tim Buckwalter. 2004. Buckwalter Arabic morphological analyzer version 2.0.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1).

Sajib Dasgupta and Vincent Ng. 2007. High-performance, language-independent morphological segmentation. In *Proceedings of Human Language Technology (NAACL)*.

Vera Demberg. 2007. A language-independent unsupervised model for morphological segmentation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, Prague, Czech Republic.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems 18*.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2007. Distributional cues to word segmentation: Context is important. In *Proceedings of the 31st Boston University Conference on Language Development*.

Alan Groves and Kirk Lowery, editors. 2006. *The Westminster Hebrew Bible Morphology Database*. Westminster Hebrew Institute, Philadelphia, PA, USA.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*.

Zellig S. Harris. 1955. From phoneme to morpheme. *Language*, 31(2):190–222.

Samarth Keshava and Emily Pitler. 2006. A simple, intuitive approach to morpheme induction. In *Proceedings of 2nd Pascal Challenges Workshop*, Venice, Italy.

Dan Klein and Christopher D. Manning. 2001. Natural language grammar induction using a constituent-context model. In *Advances in Neural Information Processing Systems 14*.

Mikko Kurimo, Mathias Creutz, and Ville Turunen. 2007. Overview of Morpho Challenge in CLEF 2007. In *Working Notes of the CLEF 2007 Workshop*.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 649–658, Honolulu, HI. ACL.

Ronald Rosenfeld. 1997. A whole sentence maximum entropy language model. In *IEEE workshop on Automatic Speech Recognition and Understanding*.

Sunita Sarawagi and William Cohen. 2004. Semimarkov conditional random fields for information extraction. In *Proceedings of the Twenty First International Conference on Machine Learning*.

Patrick Schone and Daniel Jurafsky. 2001. Knowlege-free induction of inflectional morphologies. In *Proceedings of Human Language Technology (NAACL)*.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*.

Benjamin Snyder and Regina Barzilay. 2008a. Cross-lingual propagation for morphological analysis. In *Proceedings of the Twenty Third National Conference on Artificial Intelligence*.

Benjamin Snyder and Regina Barzilay. 2008b. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

11,001 New Features for Statistical Machine Translation*

David Chiang and **Kevin Knight**
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292 USA

Wei Wang
Language Weaver, Inc.
4640 Admiralty Way, Suite 1210
Marina del Rey, CA 90292 USA

Abstract

We use the Margin Infused Relaxed Algorithm of Crammer et al. to add a large number of new features to two machine translation systems: the Hiero hierarchical phrase-based translation system and our syntax-based translation system. On a large-scale Chinese-English translation task, we obtain statistically significant improvements of +1.5 BLEU and +1.1 BLEU, respectively. We analyze the impact of the new features and the performance of the learning algorithm.

1 Introduction

What linguistic features can improve statistical machine translation (MT)? This is a fundamental question for the discipline, particularly as it pertains to improving the best systems we have. Further:

- Do syntax-based translation systems have unique and effective levers to pull when designing new features?
- Can large numbers of feature weights be learned efficiently and stably on modest amounts of data?

In this paper, we address these questions by experimenting with a large number of new features. We add more than 250 features to improve a syntax-based MT system—already the highest-scoring single system in the NIST 2008 Chinese-English common-data track—by +1.1 BLEU. We also add more than 10,000 features to Hiero (Chiang, 2005) and obtain a +1.5 BLEU improvement.

*This research was supported in part by DARPA contract HR0011-06-C-0022 under subcontract to BBN Technologies.

Many of the new features use syntactic information, and in particular depend on information that is available only inside a syntax-based translation model. Thus they widen the advantage that syntax-based models have over other types of models.

The models are trained using the Margin Infused Relaxed Algorithm or MIRA (Crammer et al., 2006) instead of the standard minimum-error-rate training or MERT algorithm (Och, 2003). Our results add to a growing body of evidence (Watanabe et al., 2007; Chiang et al., 2008) that MIRA is preferable to MERT across languages and systems, even for very large-scale tasks.

2 Related Work

The work of Och et al (2004) is perhaps the best-known study of new features and their impact on translation quality. However, it had a few shortcomings. First, it used the features for reranking n-best lists of translations, rather than for decoding or forest reranking (Huang, 2008). Second, it attempted to incorporate syntax by applying off-the-shelf part-of-speech taggers and parsers to MT output, a task these tools were never designed for. By contrast, we incorporate features directly into hierarchical and syntax-based decoders.

A third difficulty with Och et al.'s study was that it used MERT, which is not an ideal vehicle for feature exploration because it is observed not to perform well with large feature sets. Others have introduced alternative discriminative training methods (Tillmann and Zhang, 2006; Liang et al., 2006; Turian et al., 2007; Blunsom et al., 2008; Macherey et al., 2008), in which a recurring challenge is scalability: to train many features, we need many train-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 218–226,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

ing examples, and to train discriminatively, we need to search through all possible translations of each training example. Another line of research (Watanabe et al., 2007; Chiang et al., 2008) tries to squeeze as many features as possible from a relatively small dataset. We follow this approach here.

3 Systems Used

3.1 Hiero

Hiero (Chiang, 2005) is a hierarchical, string-to-string translation system. Its rules, which are extracted from unparsed, word-aligned parallel text, are synchronous CFG productions, for example:

$$\mathrm{X} \rightarrow \mathrm{X}_1 \text{ de } \mathrm{X}_2, \mathrm{X}_2 \text{ of } \mathrm{X}_1$$

As the number of nonterminals is limited to two, the grammar is equivalent to an inversion transduction grammar (Wu, 1997).

The baseline model includes 12 features whose weights are optimized using MERT. Two of the features are n-gram language models, which require intersecting the synchronous CFG with finite-state automata representing the language models. This grammar can be parsed efficiently using cube pruning (Chiang, 2007).

3.2 Syntax-based system

Our syntax-based system transforms source Chinese strings into target English syntax trees. Following previous work in statistical MT (Brown et al., 1993), we envision a noisy-channel model in which a language model generates English, and then a translation model transforms English trees into Chinese. We represent the translation model as a tree transducer (Knight and Graehl, 2005). It is obtained from bilingual text that has been word-aligned and whose English side has been syntactically parsed. From this data, we use the the GHKM minimal-rule extraction algorithm of (Galley et al., 2004) to yield rules like:

$$\text{NP-C}(x_0\text{:NPB PP(IN(of } x_1\text{:NPB))} \leftrightarrow x_1 \text{ de } x_0$$

Though this rule can be used in either direction, here we use it right-to-left (Chinese to English). We follow Galley et al. (2006) in allowing unaligned Chinese words to participate in multiple translation rules, and in collecting larger rules composed of minimal rules. These larger rules have been shown to substantially improve translation accuracy (Galley et al., 2006; DeNeefe et al., 2007).

We apply Good-Turing discounting to the transducer rule counts and obtain probability estimates:

$$P(\text{rule}) = \frac{count(\text{rule})}{count(LHS\text{-}root(\text{rule}))}$$

When we apply these probabilities to derive an English sentence e and a corresponding Chinese sentence c, we wind up with the joint probability $P(e, c)$.

The baseline model includes $\log P(e, c)$, the two n-gram language models $\log P(e)$, and other features for a total of 25. For example, there is a pair of features to punish rules that drop Chinese content words or introduce spurious English content words. All features are linearly combined and their weights are optimized using MERT.

For efficient decoding with integrated n-gram language models, all transducer rules must be binarized into rules that contain at most two variables and can be incrementally scored by the language model (Zhang et al., 2006). Then we use a CKY-style parser (Yamada and Knight, 2002; Galley et al., 2006) with cube pruning to decode new sentences.

We include two other techniques in our baseline. To get more general translation rules, we restructure our English training trees using expectation-maximization (Wang et al., 2007), and to get more specific translation rules, we relabel the trees with up to 4 specialized versions of each nonterminal symbol, again using expectation-maximization and the split/merge technique of Petrov et al. (2006).

3.3 MIRA training

We incorporate all our new features into a linear model (Och and Ney, 2002) and train them using MIRA (Crammer et al., 2006), following previous work (Watanabe et al., 2007; Chiang et al., 2008).

Let $\mathbf{e}$ stand for output strings or their derivations, and let $\mathbf{h}(\mathbf{e})$ stand for the feature vector for $\mathbf{e}$. Initialize the feature weights $\mathbf{w}$. Then, repeatedly:

- Select a batch of input sentences $\mathbf{f}_1, \ldots, \mathbf{f}_m$ and decode each $\mathbf{f}_i$ to obtain a forest of translations.
- For each i, select from the forest a set of hypothesis translations $\mathbf{e}_{i1}, \ldots, \mathbf{e}_{in}$, which are the

10-best translations according to each of:

$$\begin{gathered}\mathbf{h}(\mathbf{e}) \cdot \mathbf{w} \\ \textsc{Bleu}(\mathbf{e}) + \mathbf{h}(\mathbf{e}) \cdot \mathbf{w} \\ -\textsc{Bleu}(\mathbf{e}) + \mathbf{h}(\mathbf{e}) \cdot \mathbf{w}\end{gathered} \quad (1)$$

- For each i, select an oracle translation:

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} \left(\textsc{Bleu}(\mathbf{e}) + \mathbf{h}(\mathbf{e}) \cdot \mathbf{w}\right) \quad (2)$$

  Let $\Delta\mathbf{h}_{ij} = \mathbf{h}(\mathbf{e}_i^*) - \mathbf{h}(\mathbf{e}_{ij})$.

- For each $\mathbf{e}_{ij}$, compute the loss

$$\ell_{ij} = \textsc{Bleu}(\mathbf{e}_i^*) - \textsc{Bleu}(\mathbf{e}_{ij}) \quad (3)$$

- Update $\mathbf{w}$ to the value of $\mathbf{w}'$ that minimizes:

$$\frac{1}{2}\|\mathbf{w}' - \mathbf{w}\|^2 + C \sum_{i=1}^{m} \max_{1 \le j \le n} (\ell_{ij} - \Delta\mathbf{h}_{ij} \cdot \mathbf{w}') \quad (4)$$

  where $C = 0.01$. This minimization is performed by a variant of sequential minimal optimization (Platt, 1998).

Following Chiang et al. (2008), we calculate the sentence Bleu scores in (1), (2), and (3) in the context of some previous 1-best translations. We run 20 of these learners in parallel, and when training is finished, the weight vectors from all iterations of all learners are averaged together.

Since the interface between the trainer and the decoder is fairly simple—for each sentence, the decoder sends the trainer a forest, and the trainer returns a weight update—it is easy to use this algorithm with a variety of CKY-based decoders: here, we are using it in conjunction with both the Hiero decoder and our syntax-based decoder.

4 Features

In this section, we describe the new features introduced on top of our baseline systems.

Discount features Both of our systems calculate several features based on observed counts of rules in the training data. Though the syntax-based system uses Good-Turing discounting when computing the $P(e, c)$ feature, we find, as noted above, that it uses quite a few one-count rules, suggesting that their probabilities have been overestimated. We can directly attack this problem by adding features $\mathbf{count}_i$ that reward or punish rules seen i times, or features $\mathbf{count}_{[i,j]}$ for rules seen between i and j times.

4.1 Target-side features

String-to-tree MT offers some unique levers to pull, in terms of target-side features. Because the system outputs English trees, we can analyze output trees on the tuning set and design new features to encourage the decoder to produce more grammatical trees.

Rule overlap features While individual rules observed in decoder output are often quite reasonable, two adjacent rules can create problems. For example, a rule that has a variable of type IN (preposition) needs another rule rooted with IN to fill the position. If the second rule supplies the wrong preposition, a bad translation results. The IN node here is an *overlap point* between rules. Considering that certain nonterminal symbols may be more reliable overlap points than others, we create a binary feature for each nonterminal. A rule like:

$$\text{IN(at)} \leftrightarrow \text{zai}$$

will have feature **rule-root-IN** set to 1 and all other rule-root features set to 0. Our rule root features range over the original (non-split) nonterminal set; we have 105 in total. Even though the rule root features are locally attached to individual rules—and therefore cause no additional problems for the decoder search—they are aimed at problematic rule/rule interactions.

Bad single-level rewrites Sometimes the decoder uses questionable rules, for example:

$$\text{PP}(x_0\text{:VBN } x_1\text{:NP-C}) \leftrightarrow x_0\ x_1$$

This rule is learned from 62 cases in our training data, where the VBN is almost always the word *given*. However, the decoder misuses this rule with other VBNs. So we can add a feature that penalizes any rule in which a PP dominates a VBN and NP-C. The feature class **bad-rewrite** comprises penalties for the following configurations based on our analysis of the tuning set:

$$\begin{aligned}\text{PP} &\rightarrow \text{VBN NP-C} \\ \text{PP-BAR} &\rightarrow \text{NP-C IN} \\ \text{VP} &\rightarrow \text{NP-C PP} \\ \text{CONJP} &\rightarrow \text{RB IN}\end{aligned}$$

Node count features It is possible that the decoder creates English trees with too many or too few nodes of a particular syntactic category. For example, there may be an tendency to generate too many determiners or past-tense verbs. We therefore add a count feature for each of the 109 (non-split) English nonterminal symbols. For a rule like

$$\text{NPB(NNP(us) NNP(president) } x_0\text{:NNP)} \leftrightarrow \text{meiguo zongtong } x_0$$

the feature **node-count-NPB** gets value 1, **node-count-NNP** gets value 2, and all others get 0.

Insertion features Among the rules we extract from bilingual corpora are target-language *insertion* rules, which have a word on the English side, but no words on the source Chinese side. Sample syntax-based insertion rules are:

$$\text{NPB(DT(the) } x_0\text{:NN)} \leftrightarrow x_0$$
$$\text{S(}x_0\text{:NP-C VP(VBZ(is) } x_1\text{:VP-C))} \leftrightarrow x_0\ x_1$$

We notice that our decoder, however, frequently fails to insert words like *is* and *are*, which often have no equivalent in the Chinese source. We also notice that *the*-insertion rules sometimes have a good effect, as in the translation "in the bloom of youth," but other times have a bad effect, as in "people seek areas of the conspiracy."

Each time the decoder uses (or fails to use) an insertion rule, it incurs some risk. There is no guarantee that the interaction of the rule probabilities and the language model provides the best way to manage this risk. We therefore provide MIRA with a feature for each of the most common English words appearing in insertion rules, e.g., **insert-the** and **insert-is**. There are 35 such features.

4.2 Source-side features

We now turn to features that make use of source-side context. Although these features capture dependencies that cross boundaries between rules, they are still local in the sense that no new states need to be added to the decoder. This is because the entire source sentence, being fixed, is always available to every feature.

Soft syntactic constraints Neither of our systems uses source-side syntactic information; hence, both could potentially benefit from soft syntactic constraints as described by Marton and Resnik (2008). In brief, these features use the output of an independent syntactic parser on the source sentence, rewarding decoder constituents that match syntactic constituents and punishing decoder constituents that cross syntactic constituents. We use separately-tunable features for each syntactic category.

Structural distortion features Both of our systems have rules with variables that generalize over possible fillers, but neither system's basic model conditions a rule application on the size of a filler, making it difficult to distinguish long-distance reorderings from short-distance reorderings. To remedy this problem, Chiang et al. (2008) introduce a structural distortion model, which we include in our experiment. Our syntax-based baseline includes the generative version of this model already.

Word context During rule extraction, we retain word alignments from the training data in the extracted rules. (If a rule is observed with more than one set of word alignments, we keep only the most frequent one.) We then define, for each triple (f, e, f_{+1}), a feature that counts the number of times that f is aligned to e and f_{+1} occurs to the right of f; and similarly for triples (f, e, f_{-1}) with f_{-1} occurring to the left of f. In order to limit the size of the model, we restrict words to be among the 100 most frequently occurring words from the training data; all other words are replaced with a token <unk>.

These features are somewhat similar to features used by Watanabe et al. (2007), but more in the spirit of features used in the word sense disambiguation model introduced by Lee and Ng (2002) and incorporated as a submodel of a translation system by Chan et al. (2007); here, we are incorporating some of its features directly into the translation model.

5 Experiments

For our experiments, we used a 260 million word Chinese/English bitext. We ran GIZA++ on the entire bitext to produce IBM Model 4 word alignments, and then the link deletion algorithm (Fossum et al., 2008) to yield better-quality alignments. For

System	Training	Features	#	Tune	Test
Hiero	MERT	baseline	11	35.4	36.1
	MIRA	syntax, distortion	56	35.9	36.9*
		syntax, distortion, discount	61	36.6	37.3**
		all source-side, discount	10990	38.4	37.6**
Syntax	MERT	baseline	25	38.6	39.5
	MIRA	baseline	25	38.5	39.8*
		overlap	132	38.7	39.9*
		node count	136	38.7	40.0**
		all target-side, discount	283	39.6	40.6**

Table 1: Adding new features with MIRA significantly improves translation accuracy. Scores are case-insensitive IBM BLEU scores. $*$ or $**$ = significantly better than MERT baseline ($p < 0.05$ or 0.01, respectively).

the syntax-based system, we ran a reimplementation of the Collins parser (Collins, 1997) on the English half of the bitext to produce parse trees, then restructured and relabeled them as described in Section 3.2. Syntax-based rule extraction was performed on a 65 million word subset of the training data. For Hiero, rules with up to two nonterminals were extracted from a 38 million word subset and phrasal rules were extracted from the remainder of the training data.

We trained three 5-gram language models: one on the English half of the bitext, used by both systems, one on one billion words of English, used by the syntax-based system, and one on two billion words of English, used by Hiero. Modified Kneser-Ney smoothing (Chen and Goodman, 1998) was applied to all language models. The language models are represented using randomized data structures similar to those of Talbot et al. (2007).

Our tuning set (2010 sentences) and test set (1994 sentences) were drawn from newswire data from the NIST 2004 and 2005 evaluations and the GALE program (with no overlap at either the segment or document level). For the source-side syntax features, we used the Berkeley parser (Petrov et al., 2006) to parse the Chinese side of both sets.

We implemented the source-side context features for Hiero and the target-side syntax features for the syntax-based system, and the discount features for both. We then ran MIRA on the tuning set with 20 parallel learners for Hiero and 73 parallel learners for the syntax-based system. We chose a stopping iteration based on the BLEU score on the tuning set, and used the averaged feature weights from all iterations of all learners to decode the test set.

Syntax-based		Hiero	
count	weight	count	weight
1	+1.28	1	+2.23
2	+0.35	2	+0.77
3–5	−0.73	3	+0.54
6–10	−0.64	4	+0.29
		5+	−0.02

Table 2: Weights learned for discount features. Negative weights indicate bonuses; positive weights indicate penalties.

The results (Table 1) show significant improvements in both systems ($p < 0.01$) over already very strong MERT baselines. Adding the source-side and discount features to Hiero yields a +1.5 BLEU improvement, and adding the target-side syntax and discount features to the syntax-based system yields a +1.1 BLEU improvement. The results also show that for Hiero, the various classes of features contributed roughly equally; for the syntax-based system, we see that two of the feature classes make small contributions but time constraints unfortunately did not permit isolated testing of all feature classes.

6 Analysis

How did the various new features improve the translation quality of our two systems? We begin by examining the discount features. For these features, we used slightly different schemes for the two systems, shown in Table 2 with their learned feature weights. We see in both cases that one-count rules are strongly penalized, as expected.

Reward		Penalty	
−0.42	a	+0.67	of
−0.13	are	+0.56	the
−0.09	at	+0.47	*comma*
−0.09	on	+0.13	*period*
−0.05	was	+0.11	in
−0.05	from	+0.08	for
−0.04	's	+0.06	to
−0.04	by	+0.05	will
−0.04	is	+0.04	and
−0.03	it	+0.02	as
−0.03	its	+0.02	have
⋮		⋮	

Table 3: Weights learned for inserting target English words with rules that lack Chinese words.

6.1 Syntax features

Table 3 shows word-insertion feature weights. The system rewards insertion of forms of *be*; examples 1–3 in Figure 1 show typical improved translations that result. Among determiners, inserting *a* is rewarded, while inserting *the* is punished. This seems to be because *the* is often part of a fixed phrase, such as *the White House*, and therefore comes naturally as part of larger phrasal rules. Inserting *the* outside these fixed phrases is a risk that the generative model is too inclined to take. We also note that the system learns to punish unmotivated insertions of commas and periods, which get into our grammar via quirks in the MT training data.

Table 4 shows weights for rule-overlap features. MIRA punishes the case where rules overlap with an IN (preposition) node. This makes sense: if a rule has a variable that can be filled by any English preposition, there is a risk that an incorrect preposition will fill it. On the other hand, splitting at a period is a safe bet, and frees the model to use rules that dig deeper into NP and VP trees when constructing a top-level S. Table 5 shows weights for generated English nonterminals: SBAR-C nodes are rewarded and commas are punished.

The combined effect of all weights is subtle. To interpret them further, it helps to look at gross changes in the system's behavior. For example, a major error in the baseline system is to move "X said" or "X asked" from the beginning of the Chinese input to the middle or end of the English trans-

Bonus		Penalty	
−0.50	*period*	+0.93	IN
−0.39	VP-C	+0.57	NNP
−0.36	VB	+0.44	NN
−0.31	SG-C	+0.41	DT
−0.30	MD	+0.34	JJ
−0.26	VBG	+0.24	*right double quote*
−0.25	ADJP	+0.20	VBZ
−0.22	-LRB-	+0.19	NP
−0.21	VP-BAR	+0.16	TO
−0.20	NPB-BAR	+0.15	ADJP-BAR
−0.16	FRAG	+0.14	PRN-BAR
−0.16	PRN	+0.14	NML
−0.15	NPB	+0.13	*comma*
−0.13	RB	+0.12	VBD
−0.12	SBAR-C	+0.12	NNPS
−0.12	VP-C-BAR	+0.12	PRP
−0.11	-RRB-	+0.11	SG
⋮		⋮	

Table 4: Weights learned for employing rules whose English sides are rooted at particular syntactic categories.

Bonus		Penalty	
−0.73	SBAR-C	+1.30	*comma*
−0.54	VBZ	+0.80	DT
−0.54	IN	+0.58	PP
−0.52	NN	+0.44	TO
−0.51	PP-C	+0.33	NNP
−0.47	*right double quote*	+0.30	NNS
−0.39	ADJP	+0.30	NML
−0.34	POS	+0.22	CD
−0.31	ADVP	+0.18	PRN
−0.30	RP	+0.16	SYM
−0.29	PRT	+0.15	ADJP-BAR
−0.27	SG-C	+0.15	NP
−0.22	S-C	+0.15	MD
−0.21	NNPS	+0.15	HYPH
−0.21	VP-BAR	+0.14	PRN-BAR
−0.20	PRP	+0.14	NP-C
−0.20	NPB-BAR	+0.11	ADJP-C
⋮		⋮	

Table 5: Weights learned for generating syntactic nodes of various types anywhere in the English translation.

lation. The error occurs with many speaking verbs, and each time, we trace it to a different rule. The problematic rules can even be non-lexical, e.g.:

$$\text{S}(x_0\text{:NP-C } x_1\text{:VP } x_2\text{:},\ x_3\text{:NP-C } x_4\text{:VP } x_5\text{:.}) \leftrightarrow x_3\ x_4\ x_2\ x_0\ x_1\ x_5$$

It is therefore difficult to come up with a straightforward feature to address the problem. However, when we apply MIRA with the features already listed, these translation errors all disappear, as demonstrated by examples 4–5 in Figure 1. Why does this happen? It turns out that in translation hypotheses that move "X said" or "X asked" away from the beginning of the sentence, more commas appear, and fewer S-C and SBAR-C nodes appear. Therefore, the new features work to discourage these hypotheses. Example 6 shows additionally that commas next to speaking verbs are now correctly deleted.

Examples 7–8 in Figure 1 show other kinds of unanticipated improvements. We do not have space for a fuller analysis, but we note that the specific effects we describe above account for only part of the overall BLEU improvement.

6.2 Word context features

In Table 6 are shown feature weights learned for the word-context features. A surprising number of the highest-weighted features have to do with translations of dates and bylines. Many of the penalties seem to discourage spurious insertion or deletion of frequent words (*for*, *'s*, *said*, parentheses, and quotes). Finally, we note that several of the features (the third- and eighth-ranked reward and twelfth-ranked penalty) shape the translation of *shuo* 'said', preferring translations with an overt complementizer *that* and without a comma. Thus these features work together to attack a frequent problem that our target-syntax features also addressed.

Figure 2 shows the performance of Hiero with all of its features on the tuning and test sets over time. The scores on the tuning set rise rapidly, and the scores on the test set also rise, but much more slowly, and there appears to be slight degradation after the 18th pass through the tuning data. This seems in line with the finding of Watanabe et al. (2007) that with on the order of 10,000 features, overfitting is possible, but we can still improve accuracy on new data.

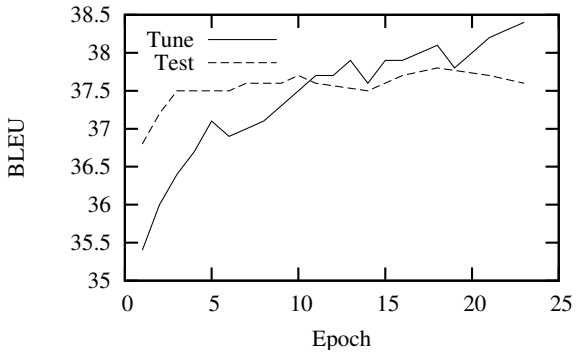


Figure 2: Using over 10,000 word-context features leads to overfitting, but its detrimental effects are modest. Scores on the tuning set were obtained from the 1-best output of the online learning algorithm, whereas scores on the test set were obtained using averaged weights.

Early stopping would have given +0.2 BLEU over the results reported in Table 1.[1]

7 Conclusion

We have described a variety of features for statistical machine translation and applied them to syntax-based and hierarchical systems. We saw that these features, discriminatively trained using MIRA, led to significant improvements, and took a closer look at the results to see how the new features qualitatively improved translation quality. We draw three conclusions from this study.

First, we have shown that these new features can improve the performance even of top-scoring MT systems. Second, these results add to a growing body of evidence that MIRA is preferable to MERT for discriminative training. When training over 10,000 features on a modest amount of data, we, like Watanabe et al. (2007), did observe overfitting, yet saw improvements on new data. Third, we have shown that syntax-based machine translation offers possibilities for features not available in other models, making syntax-based MT and MIRA an especially strong combination for future work.

[1] It was this iteration, in fact, which was used to derive the combined feature count used in the title of this paper.

1 MERT: the united states pending israeli clarification on golan settlement plan
MIRA: the united states is waiting for israeli clarification on golan settlement plan

2 MERT: . . . the average life expectancy of only 18 months , canada 's minority goverment will . . .
MIRA: . . . the average life expectancy of canada's previous minority government is only 18 months . . .

3 MERT: . . . since un inspectors expelled by north korea . . .
MIRA: . . . since un inspectors were expelled by north korea . . .

4 MERT: another thing is . . . , " he said , " obviously , the first thing we need to do . . . .
MIRA: he said : " obviously , the first thing we need to do . . . , and another thing is . . . . "

5 MERT: the actual timing . . . reopened in january , yoon said .
MIRA: yoon said the issue of the timing . . .

6 MERT: . . . us - led coalition forces , said today that the crash . . .
MIRA: . . . us - led coalition forces said today that a us military . . .

7 MERT: . . . and others will feel the danger .
MIRA: . . . and others will not feel the danger .

8 MERT: in residential or public activities within 200 meters of the region , . . .
MIRA: within 200 m of residential or public activities area , . . .

Figure 1: Improved syntax-based translations due to MIRA-trained weights.

Bonus			
	f	e	context
−1.19	<unk>	<unk>	f_{-1} = ri 'day'
−1.01	<unk>	<unk>	f_{-1} = (
−0.84	,	that	f_{-1} = shuo 'say'
−0.82	yue 'month'	<unk>	f_{+1} = <unk>
−0.78	"	"	f_{-1} = <unk>
−0.76	"	"	f_{+1} = <unk>
−0.66	<unk>	<unk>	f_{+1} = nian 'year'
−0.65	,	that	f_{+1} = <unk>
		⋮	

Penalty			
	f	e	context
+1.12	<unk>	)	f_{+1} = <unk>
+0.83	jiang 'shall'	be	f_{+1} = <unk>
+0.83	zhengfu 'government'	the	f_{-1} = <unk>
+0.73	<unk>	)	f_{-1} = <unk>
+0.73	<unk>	(	f_{+1} = <unk>
+0.72	<unk>	)	f_{-1} = ri 'day'
+0.70	<unk>	(	f_{-1} = ri 'day'
+0.69	<unk>	(	f_{-1} = <unk>
+0.66	<unk>	for	f_{-1} = <unk>
+0.66	<unk>	's	f_{-1} = ,
+0.65	<unk>	said	f_{-1} = <unk>
+0.60	,	,	f_{-1} = shuo 'say'
		⋮	

Table 6: Weights learned for word-context features, which fire when English word e is generated aligned to Chinese word f, with Chinese word f_{-1} to the left or f_{+1} to the right. Glosses for Chinese words are not part of features.

References

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proc. ACL-08: HLT*.

Peter F. Brown, Stephen A. Della Pietra, Vincent Della J. Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312.

Yee Seng Chan, Hwee Tou Ng, and David Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *Proc. ACL 2007*.

Stanley F. Chen and Joshua T. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proc. EMNLP 2008*.

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

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

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proc. ACL 1997*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proc. EMNLP-CoNLL-2007*.

Victoria Fossum, Kevin Knight, and Steven Abney. 2008. Using syntax to improve word alignment for syntax-based statistical machine translation. In *Proc. Third Workshop on Statistical Machine Translation*.

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

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

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

Kevin Knight and Jonathan Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*.

Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proc. EMNLP 2002*, pages 41–48.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. COLING-ACL 2006*.

Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakob Uskoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proc. EMNLP 2008*.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proc. ACL-08: HLT*.

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

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proc. HLT-NAACL 2004*, pages 161–168.

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

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc. ACL 2006*.

John C. Platt. 1998. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 195–208. MIT Press.

David Talbot and Miles Osborne. 2007. Randomised language modelling for statistical machine translation. In *Proc. ACL 2007*, pages 512–519.

Christoph Tillmann and Tong Zhang. 2006. A discriminative global training algorithm for statistical MT. In *Proc. COLING-ACL 2006*.

Joseph Turian, Benjamin Wellington, and I. Dan Melamed. 2007. Scalable discriminative learning for natural language parsing and translation. In *Proc. NIPS 2006*.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proc. EMNLP-CoNLL 2007*.

Taro Watanabe, Jun Suzuki, Hajime Tsukuda, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proc. EMNLP 2007*.

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

Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical MT. In *Proc. ACL 2002*.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proc. HLT-NAACL 2006*.

Efficient Parsing for Transducer Grammars

John DeNero, Mohit Bansal, Adam Pauls, and **Dan Klein**
Computer Science Division
University of California, Berkeley
{denero, mbansal, adpauls, klein}@cs.berkeley.edu

Abstract

The tree-transducer grammars that arise in current syntactic machine translation systems are large, flat, and highly lexicalized. We address the problem of parsing efficiently with such grammars in three ways. First, we present a pair of grammar transformations that admit an efficient cubic-time CKY-style parsing algorithm despite leaving most of the grammar in n-ary form. Second, we show how the number of intermediate symbols generated by this transformation can be substantially reduced through binarization choices. Finally, we describe a two-pass coarse-to-fine parsing approach that prunes the search space using predictions from a subset of the original grammar. In all, parsing time reduces by 81%. We also describe a coarse-to-fine pruning scheme for forest-based language model reranking that allows a 100-fold increase in beam size while reducing decoding time. The resulting translations improve by 1.3 BLEU.

1 Introduction

Current approaches to syntactic machine translation typically include two statistical models: a syntactic transfer model and an n-gram language model. Recent innovations have greatly improved the efficiency of language model integration through multi-pass techniques, such as forest reranking (Huang and Chiang, 2007), local search (Venugopal et al., 2007), and coarse-to-fine pruning (Petrov et al., 2008; Zhang and Gildea, 2008). Meanwhile, translation grammars have grown in complexity from simple inversion transduction grammars (Wu, 1997) to general tree-to-string transducers (Galley et al., 2004) and have increased in size by including more synchronous tree fragments (Galley et al., 2006; Marcu et al., 2006; DeNeefe et al., 2007). As a result of these trends, the syntactic component of machine translation decoding can now account for a substantial portion of total decoding time. In this paper, we focus on efficient methods for parsing with very large tree-to-string grammars, which have flat n-ary rules with many adjacent non-terminals, as in Figure 1. These grammars are sufficiently complex that the purely syntactic pass of our multi-pass decoder is the compute-time bottleneck under some conditions.

Given that parsing is well-studied in the monolingual case, it is worth asking why MT grammars are not simply like those used for syntactic analysis. There are several good reasons. The most important is that MT grammars must do both analysis and generation. To generate, it is natural to memorize larger lexical chunks, and so rules are highly lexicalized. Second, syntax diverges between languages, and each divergence expands the minimal domain of translation rules, so rules are large and flat. Finally, we see most rules very few times, so it is challenging to subcategorize non-terminals to the degree done in analytic parsing. This paper develops encodings, algorithms, and pruning strategies for such grammars.

We first investigate the qualitative properties of MT grammars, then present a sequence of parsing methods adapted to their broad characteristics. We give normal forms which are more appropriate than Chomsky normal form, leaving the rules mostly flat. We then describe a CKY-like algorithm which applies such rules efficiently, working directly over the n-ary forms in cubic time. We show how thoughtful

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 227–235,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

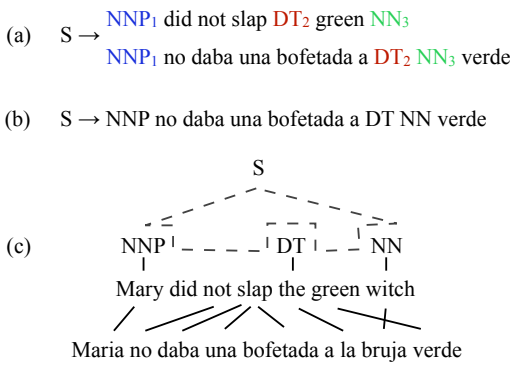


Figure 1: (a) A synchronous transducer rule has co-indexed non-terminals on the source and target side. Internal grammatical structure of the target side has been omitted. (b) The source-side projection of the rule is a monolingual source-language rule with target-side grammar symbols. (c) A training sentence pair is annotated with a target-side parse tree and a word alignment, which license this rule to be extracted.

binarization can further increase parsing speed, and we present a new coarse-to-fine scheme that uses rule subsets rather than symbol clustering to build a coarse grammar projection. These techniques reduce parsing time by 81% in aggregate. Finally, we demonstrate that we can accelerate forest-based reranking with a language model by pruning with information from the parsing pass. This approach enables a 100-fold increase in maximum beam size, improving translation quality by 1.3 BLEU while decreasing total decoding time.

2 Tree Transducer Grammars

Tree-to-string transducer grammars consist of weighted rules like the one depicted in Figure 1. Each n-ary rule consists of a root symbol, a sequence of lexical items and non-terminals on the source-side, and a fragment of a syntax tree on the target side. Each non-terminal on the source side corresponds to a unique one on the target side. Aligned non-terminals share a grammar symbol derived from a target-side monolingual grammar.

These grammars are learned from word-aligned sentence pairs annotated with target-side phrase structure trees. Extraction proceeds by using word alignments to find correspondences between target-side constituents and source-side word spans, then discovering transducer rules that match these constituent alignments (Galley et al., 2004). Given this correspondence, an array of extraction procedures yields rules that are well-suited to machine translation (Galley et al., 2006; DeNeefe et al., 2007; Marcu et al., 2006). Rule weights are estimated by discriminatively combining relative frequency counts and other rule features.

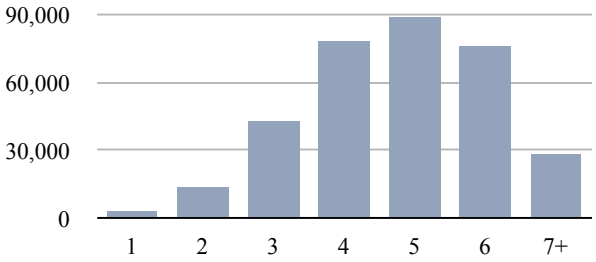


Figure 2: Transducer grammars are composed of very flat rules. Above, the histogram shows rule counts for each rule size among the 332,000 rules that apply to an individual 30-word sentence. The size of a rule is the total number of non-terminals and lexical items in its source-side yield.

A transducer grammar G can be projected onto its source language, inducing a monolingual grammar. If we weight each rule by the maximum weight of its projecting synchronous rules, then parsing with this projected grammar maximizes the translation model score for a source sentence. We need not even consider the target side of transducer rules until integrating an n-gram language model or other non-local features of the target language.

We conduct experiments with a grammar extracted from 220 million words of Arabic-English bitext, extracting rules with up to 6 non-terminals. A histogram of the size of rules applicable to a typical 30-word sentence appears in Figure 2. The grammar includes 149 grammatical symbols, an augmentation of the Penn Treebank symbol set. To evaluate, we decoded 300 sentences of up to 40 words in length from the NIST05 Arabic-English test set.

3 Efficient Grammar Encodings

Monolingual parsing with a source-projected transducer grammar is a natural first pass in multi-pass decoding. These grammars are qualitatively different from syntactic analysis grammars, such as the lexicalized grammars of Charniak (1997) or the heavily state-split grammars of Petrov et al. (2006).

In this section, we develop an appropriate grammar encoding that enables efficient parsing.

It is problematic to convert these grammars into Chomsky normal form, which CKY requires. Because transducer rules are very flat and contain specific lexical items, binarization introduces a large number of intermediate grammar symbols. Rule size and lexicalization affect parsing complexity whether the grammar is binarized explicitly (Zhang et al., 2006) or implicitly binarized using Early-style intermediate symbols (Zollmann et al., 2006). Moreover, the resulting binary rules cannot be Markovized to merge symbols, as in Klein and Manning (2003), because each rule is associated with a target-side tree that cannot be abstracted.

We also do not restrict the form of rules in the grammar, a common technique in syntactic machine translation. For instance, Zollmann et al. (2006) follow Chiang (2005) in disallowing adjacent non-terminals. Watanabe et al. (2006) limit grammars to Griebach-Normal form. However, general tree transducer grammars provide excellent translation performance (Galley et al., 2006), and so we focus on parsing with all available rules.

3.1 Lexical Normal Form

Sequences of consecutive non-terminals complicate parsing because they require a search over non-terminal boundaries when applied to a sentence span. We transform the grammar to ensure that all rules containing lexical items (lexical rules) do not contain sequences of non-terminals. We allow both unary and binary non-lexical rules.

Let L be the set of lexical items and V the set of non-terminal symbols in the original grammar. Then, *lexical normal form* (LNF) limits productions to two forms:

$$\textbf{Non-lexical: } X \rightarrow X_1(X_2)$$
$$\textbf{Lexical: } X \rightarrow (X_1)\alpha(X_2)$$
$$\alpha = w^+(X_i w^+)^*$$

Above, all $X_i \in V$ and $w^+ \in L^+$. Symbols in parentheses are optional. The nucleus α of lexical rules is a mixed sequence that has lexical items on each end and no adjacent non-terminals.

Converting a grammar into LNF requires two steps. In the *sequence elimination* step, for every lexical rule we replace each sequence of consecutive non-terminals $X_1 \ldots X_n$ with the intermediate symbol $X_1{+}\ldots{+}X_n$ (abbreviated $X_{1:n}$) and introduce a non-lexical rule $X_1{+}\ldots{+}X_n \rightarrow X_1 \ldots X_n$. In the *binarization* step, we introduce further intermediate symbols and rules to binarize all non-lexical rules in the grammar, including those added by sequence elimination.

Original grammar rules are flat and lexical

S → NNP no daba una bofetada a DT NN verde

NP → DT NN NNS

LNF replaces non-terminal sequences in lexical rules

S → NNP no daba una bofetada a DT+NN verde

DT+NN → DT NN

Non-lexical rules are binarized using few symbols

Non-lexical rules before binarization:

NP → DT NN NNS DT+NN → DT NN

Equivalent binary rules, minimizing symbol count:

NP → DT+NN NNS DT+NN → DT NN

Anchored LNF rules are bounded by lexical items

S\NNP → no daba una bofetada a DT+NN verde

NP → DT+NN NNS DT+NN → DT NN

S → NNP S\NNP

Figure 3: We transform the original grammar by first eliminating non-terminal sequences in lexical rules. Next, we binarize, adding a minimal number of intermediate grammar symbols and binary non-lexical rules. Finally, anchored LNF further transforms lexical rules to begin and end with lexical items by introducing additional symbols.

3.2 Non-terminal Binarization

Exactly how we binarize non-lexical rules affects the total number of intermediate symbols introduced by the LNF transformation.

Binarization involves selecting a set of symbols that will allow us to assemble the right-hand side $X_1 \ldots X_n$ of every non-lexical rule using binary productions. This symbol set must at least include the left-hand side of every rule in the grammar (lexical and non-lexical), including the intermediate

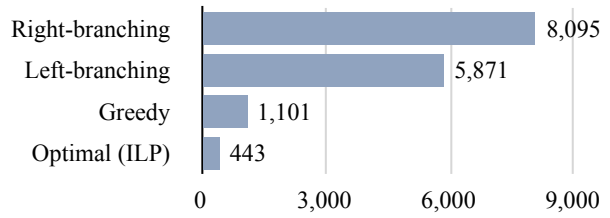


Figure 4: The number of non-terminal symbols introduced to the grammar through LNF binarization depends upon the policy for binarizing type sequences. This experiment shows results from transforming a grammar that has already been filtered for a particular short sentence. Both the greedy and optimal binarizations use far fewer symbols than naive binarizations.

symbols $X_{1:n}$ introduced by sequence elimination.

To ensure that a symbol sequence $X_1 \dots X_n$ can be constructed, we select a split point k and add intermediate types $X_{1:k}$ and $X_{k+1:n}$ to the grammar. We must also ensure that the sequences $X_1 \dots X_k$ and $X_{k+1} \dots X_n$ can be constructed. As baselines, we used *left-branching* (where $k = 1$ always) and *right-branching* (where $k = n - 1$) binarizations.

We also tested a *greedy* binarization approach, choosing k to minimize the number of grammar symbols introduced. We first try to select k such that both $X_{1:k}$ and $X_{k+1:n}$ are already in the grammar. If no such k exists, we select k such that one of the intermediate types generated is already used. If no such k exists again, we choose $k = \lfloor \frac{1}{2} n \rfloor$. This policy only creates new intermediate types when necessary. Song et al. (2008) propose a similar greedy approach to binarization that uses corpus statistics to select common types rather than explicitly reusing types that have already been introduced.

Finally, we computed an *optimal* binarization that explicitly minimizes the number of symbols in the resulting grammar. We cast the minimization as an integer linear program (ILP). Let V be the set of all base non-terminal symbols in the grammar. We introduce an indicator variable T_Y for each symbol $Y \in V^+$ to indicate that Y is used in the grammar. Y can be either a base non-terminal symbol X_i or an intermediate symbol $X_{1:n}$. We also introduce indicators $A_{Y,Z}$ for each pairs of symbols, indicating that both Y and Z are used in the grammar. Let $\mathcal{L} \subseteq V^+$ be the set of left-hand side symbols for all lexical and non-lexical rules already in the grammar. Let $\mathcal{R}$ be the set of symbol sequences on the right-hand side of all non-lexical rules. Then, the ILP takes the form:

$$\min \sum_{Y \in V^+} T_Y \quad (1)$$

$$\text{s.t. } T_Y = 1 \qquad \forall\, Y \in \mathcal{L} \quad (2)$$

$$1 \leq \sum_k A_{X_{1:k}, X_{k+1:n}} \quad \forall\, X_1 \dots X_n \in \mathcal{R} \quad (3)$$

$$T_{X_{1:n}} \leq \sum_k A_{X_{1:k}, X_{k+1:n}} \qquad \forall\, X_{1:n} \quad (4)$$

$$A_{Y,Z} \leq T_Y \;,\; A_{Y,Z} \leq T_Z \qquad \forall\, Y, Z \quad (5)$$

The solution to this ILP indicates which symbols appear in a minimal binarization. Equation 1 explicitly minimizes the number of symbols. Equation 2 ensures that all symbols already in the grammar remain in the grammar.

Equation 3 does not require that a symbol represent the entire right-hand side of each non-lexical rule, but does ensure that each right-hand side sequence can be built from two subsequence symbols. Equation 4 ensures that any included intermediate type can also be built from two subsequence types. Finally, Equation 5 ensures that if a pair is used, each member of the pair is included. This program can be optimized with an off-the-shelf ILP solver.[1]

Figure 4 shows the number of intermediate grammar symbols needed for the four binarization policies described above for a short sentence. Our ILP solver could only find optimal solutions for very short sentences (which have small grammars after relativization). Because *greedy* requires very little time to compute and generates symbol counts that are close to *optimal* when both can be computed, we use it for our remaining experiments.

3.3 Anchored Lexical Normal Form

We also consider a further grammar transformation, *anchored lexical normal form* (ALNF), in which the yield of lexical rules must begin and end with a lexical item. As shown in the following section, ALNF improves parsing performance over LNF by shifting work from lexical rule applications to non-lexical

[1] We used lp_solve: *http://sourceforge.net/projects/lpsolve*.

rule applications. ALNF consists of rules with the following two forms:

$$\textbf{Non-lexical: } X \rightarrow X_1(X_2)$$
$$\textbf{Lexical: } X \rightarrow w^+(X_i w^+)^*$$

To convert a grammar into ALNF, we first transform it into LNF, then introduce additional binary rules that split off non-terminal symbols from the ends of lexical rules, as shown in Figure 3.

4 Efficient CKY Parsing

We now describe a CKY-style parsing algorithm for grammars in LNF. The dynamic program is organized into spans S_{ij} and computes the Viterbi score $w(i, j, X)$ for each edge $S_{ij}[X]$, the weight of the maximum parse over words $i+1$ to j, rooted at symbol X. For each S_{ij}, computation proceeds in three phases: binary, lexical, and unary.

4.1 Applying Non-lexical Binary Rules

For a span S_{ij}, we first apply the binary non-lexical rules just as in standard CKY, computing an intermediate Viterbi score $w_b(i, j, X)$. Let ω_r be the weight of rule r. Then, $w_b(i, j, X) =$

$$\max_{r=X\rightarrow X_1X_2} \omega_r \max_{k=i+1}^{j-1} w(i, k, X_1) \cdot w(k, j, X_2).$$

The quantities $w(i, k, X_1)$ and $w(k, j, X_2)$ will have already been computed by the dynamic program. The work in this phase is cubic in sentence length.

4.2 Applying Lexical Rules

On the other hand, lexical rules in LNF can be applied without binarization, because they only apply to particular spans that contain the appropriate lexical items. For a given S_{ij}, we first compute all the legal mappings of each rule onto the span. A *mapping* consists of a correspondence between non-terminals in the rule and subspans of S_{ij}. In practice, there is typically only one way that a lexical rule in LNF can map onto a span, because most lexical items will appear only once in the span.

Let m be a legal mapping and r its corresponding rule. Let $S^{(i)}_{k\ell}[X]$ be the edge mapped to the ith non-terminal of r under m, and ω_r the weight of r. Then,

$$w_l(i, j, X) = \max_m \omega_r \prod_{S^{(i)}_{k\ell}[X]} w(k, \ell, X).$$

Again, $w(k, \ell, X)$ will have been computed by the dynamic program. Assuming only a constant number of mappings per rule per span, the work in this phase is quadratic. We can then merge w_l and w_b:

$$w(i, j, X) = \max(w_l(i, j, X), w_b(i, j, X)).$$

To efficiently compute mappings, we store lexical rules in a trie (or suffix array) – a searchable graph that indexes rules according to their sequence of lexical items and non-terminals. This data structure has been used similarly to index whole training sentences for efficient retrieval (Lopez, 2007). To find all rules that map onto a span, we traverse the trie using depth-first search.

4.3 Applying Unary Rules

Unary non-lexical rules are applied after lexical rules and non-lexical binary rules.

$$w(i, j, X) = \max_{r:r=X\rightarrow X_1} \omega_r w(i, j, X_1).$$

While this definition is recursive, we allow only one unary rule application per symbol X at each span to prevent infinite derivations. This choice does not limit the generality of our algorithm: chains of unaries can always be collapsed via a unary closure.

4.4 Bounding Split Points for Binary Rules

Non-lexical binary rules can in principle apply to any span S_{ij} where $j - i \geq 2$, using any split point k such that $i < k < j$. In practice, however, many rules cannot apply to many (i, k, j) triples because the symbols for their children have not been constructed successfully over the subspans S_{ik} and S_{kj}. Therefore, the precise looping order over rules and split points can influence computation time.

We found the following nested looping order for the binary phase of processing an edge $S_{ij}[X]$ gave the fastest parsing times for these grammars:

1. Loop over symbols X_1 for the left child
2. Loop over all rules $X \rightarrow X_1X_2$ containing X_1
3. Loop over split points $k : i < k < j$
4. Update $w_b(i, j, X)$ as necessary

This looping order allows for early stopping via additional bookkeeping in the algorithm. We track the following statistics as we parse:

Grammar	Bound checks	Parsing time
LNF	no	264
LNF	yes	181
ALNF	yes	104

Table 1: Adding bound checks to CKY and transforming the grammar from LNF to anchored LNF reduce parsing time by 61% for 300 sentences of length 40 or less. No approximations have been applied, so all three scenarios produce no search errors. Parsing time is in minutes.

$\min_{\text{END}}(i, X)$, $\max_{\text{END}}(i, X)$: The minimum and maximum position k for which symbol X was successfully built over S_{ik}.

$\min_{\text{START}}(j, X)$, $\max_{\text{START}}(j, X)$: The minimum and maximum position k for which symbol X was successfully built over S_{kj}.

We then bound k by $\min_k$ and $\max_k$ in the inner loop using these statistics. If ever $\min_k > \max_k$, then the loop is terminated early.

1. set $\min_k = i + 1, \max_k = j - 1$
2. loop over symbols X_1 for the left child
   $\min_k = \max(\min_k, \min_{\text{END}}(i, X_1))$
   $\max_k = \min(\max_k, \max_{\text{END}}(i, X_1))$
3. loop over rules $X \rightarrow X_1 X_2$
   $\min_k = \max(\min_k, \min_{\text{START}}(j, X_2))$
   $\max_k = \min(\max_k, \max_{\text{START}}(j, X_2))$
4. loop over split points $k : \min_k \leq k \leq \max_k$
5. update $w_b(i, j, X)$ as necessary

In this way, we eliminate unnecessary work by avoiding split points that we know beforehand cannot contribute to $w_b(i, j, X)$.

4.5 Parsing Time Results

Table 1 shows the decrease in parsing time from including these bound checks, as well as switching from lexical normal form to anchored LNF.

Using ALNF rather than LNF increases the number of grammar symbols and non-lexical binary rules, but makes parsing more efficient in three ways. First, it decreases the number of spans for which a lexical rule has a legal mapping. In this way, ALNF effectively shifts work from the lexical phase to the binary phase. Second, ALNF reduces the time spent searching the trie for mappings, because the first transition into the trie must use an edge with a lexical item. Finally, ALNF improves the frequency that, when a lexical rule matches a span, we have successfully built every edge $S_{k\ell}[X]$ in the mapping for that rule. This frequency increases from 45% to 96% with ALNF.

5 Coarse-to-Fine Search

We now consider two coarse-to-fine approximate search procedures for parsing with these grammars. Our first approach clusters grammar symbols together during the coarse parsing pass, following work in analytic parsing (Charniak and Caraballo, 1998; Petrov and Klein, 2007). We collapse all intermediate non-terminal grammar symbols (e.g., NP) to a single coarse symbol X, while pre-terminal symbols (e.g., NN) are hand-clustered into 7 classes (nouns, verbals, adjectives, punctuation, etc.). We then project the rules of the original grammar into this simplified symbol set, weighting each rule of the coarse grammar by the maximum weight of any rule that mapped onto it.

In our second and more successful approach, we select a subset of grammar symbols. We then include only and all rules that can be built using those symbols. Because the grammar includes many rules that are compositions of smaller rules, parsing with a subset of the grammar still provides meaningful scores that can be used to prune base grammar symbols while parsing under the full grammar.

5.1 Symbol Selection

To compress the grammar, we select a small subset of symbols that allow us to retain as much of the original grammar as possible. We use a voting scheme to select the symbol subset. After conversion to LNF (or ALNF), each lexical rule in the original grammar votes for the symbols that are required to build it. A rule votes as many times as it was observed in the training data to promote frequent rules. We then select the top n_l symbols by vote count and include them in the coarse grammar C.

We would also like to retain as many non-lexical rules from the original grammar as possible, but the right-hand side of each rule can be binarized in many ways. We again use voting, but this time each non-

Pruning	Minutes	Model score	BLEU
No pruning	104	60,179	44.84
Clustering	79	60,179	44.84
Subsets	50	60,163	44.82

Table 2: Coarse-to-fine pruning speeds up parsing time with minimal effect on either model score or translation quality. The coarse grammar built using symbol *subsets* outperforms *clustering* grammar symbols, reducing parsing time by 52%. These experiments do not include a language model.

lexical rule votes for its yield, a sequence of symbols. We select the top n_u symbol sequences as the set $\mathcal{R}$ of right-hand sides.

Finally, we augment the symbol set of C with intermediate symbols that can construct all sequences in $\mathcal{R}$, using only binary rules. This step again requires choosing a binarization for each sequence, such that a minimal number of additional symbols is introduced. We use the greedy approach from Section 3.2. We then include in C all rules from the original grammar that can be built from the symbols we have chosen. Surprisingly, we are able to retain 76% of the grammar rules while excluding 92% of the grammar symbols[2], which speeds up parsing substantially.

5.2 Max Marginal Thresholding

We parse first with the coarse grammar to find the Viterbi derivation score for each edge $S_{ij}[X]$. We then perform a Viterbi outside pass over the chart, like a standard outside pass but replacing $\sum$ with $\max$ (Goodman, 1999). The product of an edge's Viterbi score and its Viterbi outside score gives a *max marginal*, the score of the maximal parse that uses the edge.

We then prune away regions of the chart that deviate in their coarse max marginal from the global Viterbi score by a fixed margin tuned on a development set. Table 2 shows that both methods of constructing a coarse grammar are effective in pruning, but selecting symbol subsets outperformed the more typical clustering approach, reducing parsing time by an additional factor of 2.

[2]We used n_l of 500 and n_u of 4000 for experiments. These parameters were tuned on a development set.

6 Language Model Integration

Large n-gram language models (LMs) are critical to the performance of machine translation systems. Recent innovations have managed the complexity of LM integration using multi-pass architectures. Zhang and Gildea (2008) describes a coarse-to-fine approach that iteratively increases the order of the LM. Petrov et al. (2008) describes an additional coarse-to-fine hierarchy over language projections. Both of these approaches integrate LMs via bottom-up dynamic programs that employ beam search. As an alternative, Huang and Chiang (2007) describes a forest-based reranking algorithm called *cube growing*, which also employs beam search, but focuses computation only where necessary in a top-down pass through a parse forest.

In this section, we show that the coarse-to-fine idea of constraining each pass using marginal predictions of the previous pass also applies effectively to cube growing. Max marginal predictions from the parse can substantially reduce LM integration time.

6.1 Language Model Forest Reranking

Parsing produces a forest of derivations, where each edge in the forest holds its Viterbi (or one-best) derivation under the transducer grammar. In forest reranking via cube growing, edges in the forest produce k-best lists of derivations that are scored by both the grammar and an n-gram language model. Using ALNF, each edge must first generate a k-best list of derivations that are *not* scored by the language model. These derivations are then flattened to remove the binarization introduced by ALNF, so that the resulting derivations are each rooted by an n-ary rule r from the original grammar. The leaves of r correspond to sub-edges in the chart, which are recursively queried for their best language-model-scored derivations. These sub-derivations are combined by r, and new n-grams at the edges of these derivations are scored by the language model.

The language-model-scored derivations for the edge are placed on a priority queue. The top of the priority queue is repeatedly removed, and its successors added back on to the queue, until k language-model-scored derivations have been discovered. These k derivations are then sorted and

Pruning strategy	Max beam	BLEU	TM score	LM score	Total score	Inside time	Outside time	LM time	Total time
No pruning	20	57.67	58,570	-17,202	41,368	99	0	247	346
CTF parsing	200	58.43	58,495	-16,929	41,556	53	0	186	239
CTF reranking	200	58.63	58,582	-16,998	41,584	98	64	79	241
CTF parse + rerank	2000	58.90	58,602	-16,980	41,622	53	52	148	253

Table 3: Time in minutes and performance for 300 sentences. We used a trigram language model trained on 220 million words of English text. The *no pruning* baseline used a fix beam size for forest-based language model reranking. *Coarse-to-fine parsing* included a coarse pruning pass using a symbol subset grammar. *Coarse-to-fine reranking* used max marginals to constrain the reranking pass. *Coarse-to-fine parse + rerank* employed both of these approximations.

supplied to parent edges upon request.[3]

6.2 Coarse-to-Fine Parsing

Even with this efficient reranking algorithm, integrating a language model substantially increased decoding time and memory use. As a baseline, we reranked using a small fixed-size beam of 20 derivations at each edge. Larger beams exceeded the memory of our hardware. Results appear in Table 3.

Coarse-to-fine parsing before LM integration substantially improved language model reranking time. By pruning the chart with max marginals from the coarse symbol subset grammar from Section 5, we were able to rerank with beams of length 200, leading to a 0.8 BLEU increase and a 31% reduction in total decoding time.

6.3 Coarse-to-Fine Forest Reranking

We realized similar performance and speed benefits by instead pruning with max marginals from the full grammar. We found that LM reranking explored many edges with low max marginals, but used few of them in the final decoder output. Following the coarse-to-fine paradigm, we restricted the reranker to edges with a max marginal above a fixed threshold. Furthermore, we varied the beam size of each edge based on the parse. Let Δ_m be the ratio of the max marginal for edge m to the global Viterbi derivation for the sentence. We used a beam of size $\lceil k \cdot 2^{\ln \Delta_m} \rceil$ for each edge.

Computing max marginals under the full grammar required an additional outside pass over the full parse forest, adding substantially to parsing time. However, soft coarse-to-fine pruning based on these max marginals also allowed for beams up to length 200, yielding a 1.0 BLEU increase over the baseline and a 30% reduction in total decoding time.

We also combined the coarse-to-fine parsing approach with this soft coarse-to-fine reranker. Tiling these approximate search methods allowed another 10-fold increase in beam size, further improving BLEU while only slightly increasing decoding time.

7 Conclusion

As translation grammars increase in complexity while innovations drive down the computational cost of language model integration, the efficiency of the parsing phase of machine translation decoding is becoming increasingly important. Our grammar normal form, CKY improvements, and symbol subset coarse-to-fine procedure reduced parsing time for large transducer grammars by 81%.

These techniques also improved forest-based language model reranking. A full decoding pass without any of our innovations required 511 minutes using only small beams. Coarse-to-fine pruning in both the parsing and language model passes allowed a 100-fold increase in beam size, giving a performance improvement of 1.3 BLEU while decreasing total decoding time by 50%.

Acknowledgements

This work was enabled by the Information Sciences Institute Natural Language Group, primarily through the invaluable assistance of Jens Voeckler, and was supported by the National Science Foundation (NSF) under grant IIS-0643742.

[3] Huang and Chiang (2007) describes the cube growing algorithm in further detail, including the precise form of the successor function for derivations.

References

Eugene Charniak and Sharon Caraballo. 1998. New figures of merit for best-first probabilistic chart parsing. In *Computational Linguistics*.

Eugene Charniak. 1997. Statistical techniques for natural language parsing. In *National Conference on Artificial Intelligence*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *The Annual Conference of the Association for Computational Linguistics*.

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *The Annual Conference of the Association for Computational Linguistics*.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *The Annual Conference of the Association for Computational Linguistics*.

Dan Klein and Chris Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the Association for Computational Linguistics*.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *The Conference on Empirical Methods in Natural Language Processing*.

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

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *The Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *The Annual Conference of the Association for Computational Linguistics*.

Slav Petrov, Aria Haghighi, and Dan Klein. 2008. Coarse-to-fine syntactic machine translation using language projections. In *The Conference on Empirical Methods in Natural Language Processing*.

Xinying Song, Shilin Ding, and Chin-Yew Lin. 2008. Better binarization for the CKY parsing. In *The Conference on Empirical Methods in Natural Language Processing*.

Ashish Venugopal, Andreas Zollmann, and Stephan Vogel. 2007. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *In Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference*.

Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *The Annual Conference of the Association for Computational Linguistics*.

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

Hao Zhang and Daniel Gildea. 2008. Efficient multi-pass decoding for synchronous context free grammars. In *The Annual Conference of the Association for Computational Linguistics*.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *North American Chapter of the Association for Computational Linguistics*.

Andreas Zollmann, Ashish Venugopal, and Stephan Vogel. 2006. Syntax augmented machine translation via chart parsing. In *The Statistical Machine Translation Workshop at the North American Association for Computational Linguistics Conference*.

Preference Grammars: Softening Syntactic Constraints to Improve Statistical Machine Translation

Ashish Venugopal Andreas Zollmann Noah A. Smith Stephan Vogel
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{ashishv,zollmann,nasmith,vogel}@cs.cmu.edu

Abstract

We propose a novel probabilistic synchoronous context-free grammar formalism for statistical machine translation, in which syntactic nonterminal labels are represented as "soft" preferences rather than as "hard" matching constraints. This formalism allows us to efficiently score *unlabeled* synchronous derivations without forgoing traditional syntactic constraints. Using this score as a feature in a log-linear model, we are able to approximate the selection of the most likely unlabeled derivation. This helps reduce fragmentation of probability across differently labeled derivations of the same translation. It also allows the importance of syntactic preferences to be learned alongside other features (e.g., the language model) and for particular labeling procedures. We show improvements in translation quality on small and medium sized Chinese-to-English translation tasks.

1 Introduction

Probabilistic synchronous context-free grammars (PSCFGs) define weighted production rules that are automatically learned from parallel training data. As in classical CFGs, these rules make use of nonterminal symbols to generalize beyond lexical modeling of sentences. In MT, this permits translation and reordering to be conditioned on more abstract notions of context. For example,

$$\text{VP} \rightarrow \text{ne VB}_1 \text{ pas \# do not VB}_1$$

represents the discontiguous translation of the French words "ne" and "pas" to "do not", in the context of the *labeled* nonterminal symbol "VB" (representing syntactic category "verb"). Translation with PSCFGs is typically expressed as the problem of finding the maximum-weighted derivation consistent with the source sentence, where the scores are defined (at least in part) by $\mathbb{R}$-valued weights associated with the rules. A PSCFG derivation is a synchronous parse tree.

Defining the translation function as finding the best derivation has the unfortunate side effect of forcing differently-derived versions of the same target sentence to compete with each other. In other words, the true score of each translation is "fragmented" across many derivations, so that each translation's most probable derivation is the only one that matters. The more Bayesian approach of finding the most probable *translation* (integrating out the derivations) instantiates an NP-hard inference problem even for simple word-based models (Knight, 1999); for grammar-based translation it is known as the consensus problem (Casacuberta and de la Higuera, 2000; Sima'an, 2002).

With weights interpreted as probabilities, the maximum-weighted derivation is the maximum *a posteriori* (MAP) derivation:

$$\hat{e} \leftarrow \operatorname*{argmax}_e \max_d p(e, d \mid f)$$

where f is the source sentence, e ranges over target sentences, and d ranges over PSCFG derivations (synchronous trees). This is often described as an approximation to the most probable translation, $\operatorname{argmax}_e \sum_d p(e, d \mid f)$. In this paper, we will describe a technique that aims to find the most probable *equivalence class* of *unlabeled* derivations, rather than a single labeled derivation, reducing the fragmentation problem. Solving this problem exactly is still an NP-hard consensus problem, but we provide approximations that build on well-known PSCFG decoding methods. Our model falls somewhere between PSCFGs that extract nonterminal symbols from parse trees and treat them as part of

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 236–244,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

the derivation (Zollmann and Venugopal, 2006) and unlabeled hierarchical structures (Chiang, 2005); we treat nonterminal labels as random variables chosen at each node, with each (unlabeled) rule expressing "preferences" for particular nonterminal labels, learned from data.

The paper is organized as follows. In Section 2, we summarize the use of PSCFG grammars for translation. We describe our model (Section 3). Section 4 explains the preference-related calculations, and Section 5 addresses decoding. Experimental results using preference grammars in a log-linear translation model are presented for two standard Chinese-to-English tasks in Section 6. We review related work (Section 7) and conclude.

2 PSCFGs for Machine Translation

Probabilistic synchronous context-free grammars (PSCFGs) are defined by a source terminal set (source vocabulary) $\mathcal{T}_S$, a target terminal set (target vocabulary) $\mathcal{T}_T$, a shared nonterminal set $\mathcal{N}$ and a set $\mathcal{R}$ of rules of the form: $X \rightarrow \langle \gamma, \alpha, w \rangle$ where

- $X \in \mathcal{N}$ is a labeled nonterminal referred to as the left-hand-side of the rule.
- $\gamma \in (\mathcal{N} \cup \mathcal{T}_S)^*$ is the source side of the rule.
- $\alpha \in (\mathcal{N} \cup \mathcal{T}_T)^*$ is the target side of the rule.
- $w \in [0, \infty)$ is a nonnegative real-valued weight assigned to the rule.

For visual clarity, we will use the # character to separate the source side of the rule γ from the target side α. PSCFG rules also have an implied one-to-one mapping between nonterminal symbols in γ and nonterminals symbols in α. Chiang (2005), Zollmann and Venugopal (2006) and Galley et al. (2006) all use parameterizations of this PSCFG formalism[1].

Given a source sentence f and a PSCFG G, the translation task can be expressed similarly to monolingual parsing with a PCFG. We aim to find the most likely derivation d of the input source sentence and read off the English translation, identified by composing α from each rule used in the derivation. This search for the most likely translation under the MAP approximation can be defined as:

$$\hat{e} = \mathrm{tgt}\left(\underset{d \in \mathcal{D}(G):\mathrm{src}(d)=f}{\mathrm{argmax}} p(d)\right) \quad (1)$$

where $\mathrm{tgt}(d)$ is the target-side yield of a derivation d, and $\mathcal{D}(G)$ is the set of G's derivations. Using an n-gram language model to score derivations and rule labels to constraint the rules that form derivations, we define $p(d)$ as log-linear model in terms of the rules $r \in \mathcal{R}$ used in d as:

$$\begin{aligned} p(d) &= p_{\mathrm{LM}}(\mathrm{tgt}(d))^{\lambda_0} \times \left(\prod_{i=1}^{m} p_i(d)^{\lambda_i}\right) \\ &\quad \times p_{\mathrm{syn}}(d)^{\lambda_{m+1}} / Z(\vec{\lambda}) \\ p_i(d) &= \prod_{r \in \mathcal{R}} h_i(r)^{\mathrm{freq}(r;d)} \quad (2) \\ p_{\mathrm{syn}}(d) &= \begin{cases} 1 & \text{if } d \text{ respects label constraints} \\ 0 & \text{otherwise} \end{cases} \quad (3) \end{aligned}$$

where $\vec{\lambda} = \lambda_0 \cdots \lambda_{m+1}$ are weights that reflect the relative importance of features in the model. The features include the n-gram language model (LM) score of the target yield sequence, a collection of m rule feature functions $h_i : \mathcal{R} \rightarrow \mathbb{R}_{\geq 0}$, and a "syntax" feature that (redundantly) requires every nonterminal token to be expanded by a rule with that nonterminal on its left-hand side. $\mathrm{freq}(r; d)$ denotes the frequency of the rule r in the derivation d. Note that λ_{m+1} can be effectively ignored when p_{syn} is defined as in Equation 3. $Z(\vec{\lambda})$ is a normalization constant that does not need to be computed during search under the argmax search criterion in Equation 1. Feature weights $\vec{\lambda}$ are trained discriminatively in concert with the language model weight to maximize the BLEU (Papineni et al., 2002) automatic evaluation metric via Minimum Error Rate Training (MERT) (Och, 2003).

We use the open-source PSCFG rule extraction framework and decoder from Zollmann et al. (2008) as the framework for our experiments. The asymptotic runtime of this decoder is:

$$\mathcal{O}\left(|f|^3 \left[|\mathcal{N}||\mathcal{T}_T|^{2(n-1)}\right]^K\right) \quad (4)$$

where K is the maximum number of nonterminal symbols per rule, $|f|$ the source sentence length, and

[1] Galley et al. (2006) rules are formally defined as tree transducers but have equivalent PSCFG forms.

n is the order of the n-gram LM that is used to compute p_{LM}. This constant factor in Equation 4 arises from the dynamic programming item structure used to perform search under this model. Using notation from Chiang (2007), the corresponding item structure is:

$$[X, i, j, q(\alpha)] : w \tag{5}$$

where X is the nonterminal label of a derivation, i, j define a span in the source sentence, and $q(\alpha)$ maintains state required to compute $p_{\text{LM}}(\alpha)$. Under the MAP criterion we can discard derivations of lower weight that share this item structure, but in practice we often require additional lossy pruning to limit the number of items produced. The Syntax-Augmented MT model of Zollmann and Venugopal (2006), for instance, produces a very large nonterminal set using "slash" (NP/NN → *the great*) and "plus" labels (NP+VB → *she went*) to assign syntactically motivated labels for rules whose target words do not correspond to constituents in phrase structure parse trees. These labels lead to fragmentation of probability across many derivations for the same target sentence, worsening the impact of the MAP approximation. In this work we address the increased fragmentation resulting from rules with labeled nonterminals compared to unlabeled rules (Chiang, 2005).

3 Preference Grammars

We extend the PSCFG formalism to include soft "label preferences" for unlabeled rules that correspond to alternative labelings that have been encountered in training data for the unlabeled rule form. These preferences, estimated via relative frequency counts from rule occurrence data, are used to estimate the feature $p_{\text{syn}}(d)$, the probability that an unlabeled derivation can be generated under traditional syntactic constraints. In classic PSCFG, $p_{\text{syn}}(d)$ enforces a hard syntactic constraint (Equation 3). In our approach, label preferences influence the value of $p_{\text{syn}}(d)$.

3.1 Motivating example

Consider the following labeled Chinese-to-English PSCFG rules:

(4) S → 我 在 哪 能 VB_1 #
a place where I can VB_1

(3) S → 我 在 哪 能 VP_1 #
a place where I can VP_1

(2) SBAR → 我 在 哪 能 VP_1 #
a place where I can VP_1

(1) FRAG → 我 在 哪 能 AUX_1 #
a place where I can AUX_1

(8) VB → 吃饭 # eat

(1) VP → 吃饭 # eat

(1) NP → 吃饭 # eat

(10) NN → 吃饭 # dish

where the numbers are frequencies of the rule from the training corpus. In classical PSCFG we can think of the nonterminals mentioned in the rules as hard constraints on which rules can be used to expand a particular node; e.g., a VP can only be expanded by a VP rule. In Equation 2, $p_{\text{syn}}(d)$ explicitly enforces this hard constraint. Instead, we propose softening these constraints. In the rules below, labels are represented as soft preferences.

(10) X → 我 在 哪 能 X_1 #
a place where I can X_1

$$\left\{ \begin{array}{rcl} p(H_0 = \text{S}, H_1 = \text{VB} \mid r) & = & 0.4 \\ p(H_0 = \text{S}, H_1 = \text{VP} \mid r) & = & 0.3 \\ p(H_0 = \text{SBAR}, H_1 = \text{VP} \mid r) & = & 0.2 \\ p(H_0 = \text{FRAG}, H_1 = \text{AUX} \mid r) & = & 0.1 \end{array} \right\}$$

(10) X → 吃饭 # eat

$$\left\{ \begin{array}{rcl} p(H_0 = \text{VB} \mid r) & = & 0.8 \\ p(H_0 = \text{VP} \mid r) & = & 0.1 \\ p(H_0 = \text{NP} \mid r) & = & 0.1 \end{array} \right\}$$

(10) X → 吃饭 # dish

$$\left\{ \; p(H_0 = \text{NN} \mid r) \quad = \quad 1.0 \; \right\}$$

Each unlabeled form of the rule has an associated distribution over labels for the nonterminals referenced in the rule; the labels are random variables H_i, with H_0 the left-hand-side label. These unlabeled rule forms are simply packed representations of the original labeled PSCFG rules. In addition to the usual features $h_i(r)$ for each rule, estimated based on unlabeled rule frequencies, we now

have label preference distributions. These are estimated as relative frequencies from the labelings of the base, unlabeled rule. Our primary contribution is how we compute $p_{\text{syn}}(d)$—the probability that an unlabeled derivation adheres to traditional syntactic constraints—for derivations built from preference grammar rules. By using $p_{\text{syn}}(d)$ as a feature in the log-linear model, we allow the MERT framework to evaluate the importance of syntactic structure relative to other features.

The example rules above highlight the potential for $p_{\text{syn}}(d)$ to affect the choice of translation. The translation of the Chinese word sequence 我 在 哪 能 吃饭 can be performed by expanding the nonterminal in the rule "a place where I can X_1" with either "eat" or "dish." A hierarchical system (Chiang, 2005) would allow either expansion, relying on features like p_{LM} to select the best translation since both expansions occurred the same number of times in the data.

A richly-labeled PSCFG as in Zollmann and Venugopal (2006) would immediately reject the rule generating "dish" due to hard label matching constraints, but would produce three identical, competing derivations. Two of these derivations would produce S as a root symbol, while one derivation would produce SBAR. The two S-labeled derivations compete, rather than reinforce the choice of the word "eat," which they both make. They will also compete for consideration by any decoder that prunes derivations to keep runtime down.

The rule preferences indicate that VB and VP are both valid labels for the rule translating to "eat", and both of these labels are compatible with the arguments expected by "a place where I can X_1". Alternatively, "dish" produces a NN label which is not compatible with the arguments of this higher-up rule. We design $p_{\text{syn}}(d)$ to reflect compatibility between two rules (one expanding a right-hand side nonterminal in the other), based on label preference distributions.

3.2 Formal definition

Probabilistic synchronous context-free *preference* grammars are defined as PSCFGs with the following additional elements:

- $\mathcal{H}$: a set of implicit labels, not to be confused with the *explicit* label set $\mathcal{N}$.
- π: $\mathcal{H} \rightarrow \mathcal{N}$, a function that associates each implicit label with a single explicit label. We can therefore think of $\mathcal{H}$ symbols as refinements of the nonterminals in $\mathcal{N}$ (Matsusaki et al., 2005).
- For each rule r, we define a probability distribution over vectors $\vec{h}$ of implicit label bindings for its nonterminals, denoted $p_{\text{pref}}(\vec{h} \mid r)$. $\vec{h}$ includes bindings for the left-hand side nonterminal (h_0) as well as each right-hand side nonterminal ($h_1, ..., h_{|\vec{h}|}$). Each $h_i \in \mathcal{H}$.

When $\mathcal{N}, \mathcal{H}$ are defined to include just a single generic symbol as in (Chiang, 2005), we produce the unlabeled grammar discussed above. In this work, we define

- $\mathcal{N} = \{S, X\}$
- $\mathcal{H} = \{\text{NP}, \text{DT}, \text{NN} \cdots\} = \mathcal{N}_{\text{SAMT}}$

where $\mathcal{N}$ corresponds to the generic labels of Chiang (2005) and $\mathcal{H}$ corresponds to the syntactically motivated SAMT labels from (Zollmann and Venugopal, 2006), and π maps all elements of $\mathcal{H}$ to X. We will use $\text{hargs}(r)$ to denote the set of all $\vec{h} = \langle h_0, h_1, ..., h_k \rangle \in \mathcal{H}^{k+1}$ that are valid bindings for the rule with nonzero preference probability.

The preference distributions p_{pref} from each rule used in d are used to compute $p_{\text{syn}}(d)$ as described next.

4 Computing feature $p_{\text{syn}}(d)$

Let us view a derivation d as a collection of nonterminal tokens n_j, $j \in \{1, ..., |d|\}$. Each n_j takes an explicit label in $\mathcal{N}$. The score $p_{\text{syn}}(d)$ is a product, with one factor per n_j in the derivation d:

$$p_{\text{syn}}(d) = \prod_{j=1}^{|d|} \phi_j \tag{6}$$

Each ϕ_j factor considers the two rules that n_j participates in. We will refer to the rule above nonterminal token n_j as $\underline{r}_j$ (the nonterminal is a child in this rule) and the rule that expands nonterminal token j as $\overline{r}_j$.

The intuition is that derivations in which these two rules agree (at each j) about the implicit label

for n_j, in $\mathcal{H}$ are preferable to derivations in which they do not. Rather than making a decision about the implicit label, we want to reward p_{syn} when $\underline{r}_j$ and $\underline{r}_j$ are consistent. Our way of measuring this consistency is an inner product of preference distributions:

$$\phi_j \propto \sum_{h \in \mathcal{H}} p_{\text{pref}}(h \mid \underline{r}_j) p_{\text{pref}}(h \mid \overline{r}_j) \tag{7}$$

This is not quite the whole story, because $p_{\text{pref}}(\cdot \mid r)$ is defined as a joint distribution of *all* the implicit labels within a rule; the implicit labels are not independent of each other. Indeed, we want the implicit labels within each rule to be mutually consistent, i.e., to correspond to one of the rule's preferred labelings, for both $\text{hargs}(\underline{r})$ and $\text{hargs}(\overline{r})$.

Our approach to calculating p_{syn} within the dynamic programming algorithm is to recursively calculate preferences for each chart item based on (a) the smaller items used to construct the item and (b) the rule that permits combination of the smaller items into the larger one. We describe how the preferences for chart items are calculated. Let a chart item be denoted $[X, i, j, u, ...]$ where $X \in \mathcal{N}$ and i and j are positions in the source sentence, and

$$u : \{h \in \mathcal{H} \mid \pi(h) = X\} \rightarrow [0, 1]$$

(where $\sum_h u(h) = 1$) denotes a distribution over possible X-refinement labels. We will refer to it below as the *left-hand-side preference distribution*. Additional information (such as language model state) may also be included; it is not relevant here.

The simplest case is for a nonterminal token n_j that has no nonterminal children. Here the left-hand-side preference distribution is simply given by

$$u(h) = p_{\text{pref}}(h \mid \overline{r}_j) \ .$$

and we define the p_{syn} factor to be $\phi_j = 1$.

Now consider the dynamic programming step of combining an already-built item $[X, i, j, u, ...]$ rooted by explicit nonterminal X, spanning source sentence positions i to j, with left-hand-side preference distribution u, to build a larger item rooted by Y through a rule $r = Y \rightarrow \langle \gamma X_1 \gamma', \alpha X_1 \alpha', w \rangle$ with preferences $p_{\text{pref}}(\cdot \mid r)$.[2] The new item will have signature $[Y, i - |\gamma|, j + |\gamma'|, v, ...]$. The left-hand-side preferences v for the new item are calculated as follows:

$$v(h) = \frac{\tilde{v}(h)}{\sum_{h'} \tilde{v}(h')} \quad \text{where} \tag{8}$$

$$\tilde{v}(h) = \sum_{h' \in \mathcal{H} : \langle h, h' \rangle \in \text{hargs}(r)} p_{\text{pref}}(\langle h, h' \rangle \mid r) \times u(h')$$

Renormalizing keeps the preference vectors on the same scale as those in the rules. The p_{syn} factor ϕ, which is factored into the value of the new item, is calculated as:

$$\phi = \sum_{h' \in \mathcal{H} : \langle h, h' \rangle \in \text{hargs}(r)} u(h') \tag{9}$$

so that the value considered for the new item is $w \times \phi \times ...$, where factors relating to p_{LM}, for example, may also be included. Coming back to our example, if we let $\overline{r}$ be the leaf rule producing "eat" at shared nonterminal n_1, we generate an item with:

$$u = \langle u(\text{VB}) = 0.8, u(\text{VP}) = 0.1, u(\text{NP}) = 0.1 \rangle$$
$$\phi_1 = 1$$

Combining this item with X → ⟨ 我 在 哪 能 X_1 # a place where I can X_1 ⟩ as $\underline{r}_2$ at nonterminal n_2 generates a new target item with translation "a place where I can eat", $\phi_2 = 0.9$ and v as calculated in Fig. 1. In contrast, $\phi_2 = 0$ for the derivation where $\overline{r}$ is the leaf rule that produces "dish".

This calculation can be seen as a kind of single-pass, bottom-up message passing inference method embedded within the usual dynamic programming search.

5 Decoding Approximations

As defined above, accurately computing $p_{\text{syn}}(d)$ requires extending the chart item structure with u. For models that use the n-gram LM feature, the item structure would be:

$$[X, i, j, q(\alpha), u] : w \tag{10}$$

Since u effectively summarizes the choice of rules in a derivation, this extension would partition the

[2] We assume for the discussion that $\alpha, \alpha' \in \mathcal{T}_S^*$ and $\gamma, \gamma' \in \mathcal{T}_T^*$. If there are multiple nonterminals on the right-hand side of the rule, we sum over the longer sequences in $\text{hargs}(r)$ and include appropriate values from the additional "child" items' preference vectors in the product.

$\tilde{v}(\text{S}) = p_{pref}(\langle h = \text{S}, h' = \text{VB}\rangle \mid \underline{r})u(\text{VB}) + p_{pref}(\langle h = \text{S}, h' = \text{VP}\rangle \mid \underline{r})u(\text{VP}) = (0.4 \times 0.8) + (0.3 \times 0.1) = 0.35$
$\tilde{v}(\text{SBAR}) = p(\langle h = \text{SBAR}, h' = \text{VP}\rangle \mid \underline{r})u(\text{VP}) = (0.2 \times 0.1) = 0.02$
$v = \langle v(\text{S}) = 0.35/(\tilde{v}(\text{S}) + \tilde{v}(\text{SBAR})), v(\text{SBAR}) = 0.02/\tilde{v}(\text{S}) + \tilde{v}(\text{SBAR})\rangle = \langle v(\text{S}) = 0.35/0.37, v(\text{SBAR}) = 0.02/0.37\rangle$
$\phi_2 = u(\text{VB}) + u(\text{VP}) = 0.8 + 0.1 = 0.9$

Figure 1: Calculating v and ϕ_2 for the running example.

search space *further*. To prevent this partitioning, we follow the approach of Venugopal et al. (2007). We keep track of u for the best performing derivation from the set of derivations that share $[X, i, j, q(\alpha)]$ in a first-pass decoding. In a second top-down pass similar to Huang and Chiang (2007), we can recalculate $p_{syn}(d)$ for alternative derivations in the hypergraph; potentially correcting search errors made in the first pass.

We face another significant practical challenge during decoding. In real data conditions, the size of the preference vector for a single rule can be very high, especially for rules that include multiple nonterminal symbols that are located on the left and right boundaries of γ. For example, the Chinese-to-English rule X $\rightarrow$ ⟨ X$_1$ 的 X$_2$ # X$_1$'s X$_2$ ⟩ has over 24K elements in $\text{hargs}(r)$ when learned for the medium-sized NIST task used below. In order to limit the explosive growth of nonterminals during decoding for both memory and runtime reasons, we define the following label pruning parameters:

- β_R: This parameter limits the size of $\text{hargs}(r)$ to the β_R top-scoring preferences, defaulting other values to zero.
- β_L: This parameter is the same as β_R but applied only to rules with no nonterminals. The stricter of β_L and β_R is applied if both thresholds apply.
- β_P: This parameter limits the number labels in *item* preference vectors (Equation 8) to the β_P most likely labels during decoding, defaulting other preferences to zero.

6 Empirical Results

We evaluate our preference grammar model on small (IWSLT) and medium (NIST) data Chinese-to-English translation tasks (described in Table 1). IWSLT is a limited domain, limited resource task (Paul, 2006), while NIST is a broadcast news task with wide genre and domain coverage. We use a subset of the full training data (67M words of English text) from the annual NIST MT Evaluation. Development corpora are used to train model parameters via MERT. We use a variant of MERT that prefers sparse solutions where $\lambda_i = 0$ for as many features as possible. At each MERT iteration, a subset of features λ are assigned 0 weight and optimization is repeated. If the resulting BLEU score is not lower, these features are left at zero.

All systems are built on the SAMT framework described in Zollmann et al. (2008), using a trigram LM during search and the full-order LM during a second hypergraph rescoring pass. Reordering limits are set to 10 words for all systems. Pruning parameters during decoding limit the number of derivations at each source span to 300.

The system "Hier." uses a grammar with a single nonterminal label as in Chiang (2005). The system "Syntax" applies the grammar from Zollmann and Venugopal (2006) that generates a large number of syntactically motivated nonterminal labels. For the NIST task, rare rules are discarded based on their frequency in the training data. Purely lexical rules (that include no terminal symbols) that occur less than 2 times, or non-lexical rules that occur less than 4 times are discarded.

IWSLT task: We evaluate the preference grammar system "Pref." with parameters $\beta_R = 100$, $\beta_L = 5$, $\beta_P = 2$. Results comparing systems Pref. to Hier. and Syntax are shown in Table 2. Automatic evaluation results using the preference grammar translation model are positive. The preference grammar system shows improvements over both the Hier. and Syntax based systems on both unseen evaluation sets IWSLT 2007 and 2008. The improvements are clearest on the BLEU metric (matching the MERT training criteria). On 2007 test data, Pref. shows a 1.2-point improvement over Hier., while on the 2008 data, there is a 0.6-point improvement. For the IWSLT task, we report additional au-

System Name	Words in Target Text	LM singleton 1-n-grams (n)	Dev.	Test
IWSLT	632K	431K (5)	IWSLT06	IWSLT07,08
NIST	67M	102M (4)	MT05	MT06

Table 1: Training data configurations used to evaluate preference grammars. The number of words in the target text and the number of singleton 1-n-grams represented in the complete model are the defining statistics that characterize the scale of each task. For each LM we also indicate the order of the n-gram model.

System	Dev BLEU (lpen) ↑	2007 BLEU (lpen) ↑	2008 BLEU (lpen) ↑	2008 WER ↓	2008 PER ↓	2008 MET. ↑	2008 GTM ↑
Hier.	28.0 (0.89)	37.0 (0.89)	45.9 (0.91)	44.5	39.9	61.8	70.7
Syntax	30.9 (0.96)	35.5 (0.94)	45.3 (0.95)	45.7	40.4	62.1	71.5
Pref.	28.3 (0.88)	38.2 (0.90)	46.3 (0.91)	43.8	40.0	61.7	71.2

Table 2: Translation quality metrics on the IWSLT translation task, with IWSLT 2006 as the development corpora, and IWSLT 2007 and 2008 as test corpora. Each metric is annotated with an ↑ if increases in the metric value correspond to increase in translation quality and a ↓ if the opposite is true. We also list length penalties for the BLEU metric to show that improvements are not due to length optimizations alone.

tomatic evaluation metrics that generally rank the Pref. system higher than Hier. and Syntax. As a further confirmation, our feature selection based MERT chooses to retain λ_{m+1} in the model. While the IWSLT results are promising, we perform a more complete evaluation on the NIST translation task.

NIST task: This task generates much larger rule preference vectors than the IWSLT task simply due to the size of the training corpora. We build systems with both $\beta_R = 100, 10$ varying β_P. Varying β_P isolates the relative impact of propagating alternative nonterminal labels within the preference grammar model. $\beta_L = 5$ for all NIST systems. Parameters λ are trained via MERT on the $\beta_R = 100$, $\beta_L = 5$, $\beta_P = 2$ system. BLEU scores for each preference grammar and baseline system are shown in Table 3, along with translation times on the test corpus. We also report length penalties to show that improvements are not simply due to better tuning of output length.

The preference grammar systems outperform the Hier. baseline by 0.5 points on development data, and upto 0.8 points on unseen test data. While systems with $\beta_R = 100$ take significantly longer to translate the test data than Hier., setting $\beta_R = 10$ takes approximately as long as the Syntax based system but produces better slightly better results (0.3 points).

The improvements in translation quality with the preference grammar are encouraging, but how much of this improvement can simply be attributed to MERT finding a better local optimum for parameters λ? To answer this question, we use parameters $\lambda^\star$ optimized by MERT for the preference grammar system to run a purely hierarchical system, denoted Hier.($\lambda^\star$), which ignores the value of λ_{m+1} during decoding. While almost half of the improvement comes from better parameters learned via MERT for the preference grammar systems, 0.5 points can be still be attributed purely to the feature p_{syn}. In addition, MERT does not set parameter λ_{m+1} to 0, corroborating the value of the p_{syn} feature again. Note that Hier.($\lambda^\star$) achieves better scores than the Hier. system which was trained via MERT without p_{syn}. This highlights the local nature of MERT parameter search, but also points to the possibility that training with the feature p_{syn} produced a more diverse derivation space, resulting in better parameters λ. We see a very small improvement (0.1 point) by allowing the runtime propagation of more than 1 nonterminal label in the left-hand side posterior distribution, but the improvement doesn't extend to $\beta_P = 5$. Improved integration of the feature $p_{syn}(d)$ into decoding might help to widen this gap.

System	Dev. BLEU (lpen)	Test BLEU (lpen)	Test time (h:mm)
Baseline Systems			
Hier.	34.1 (0.99)	31.8 (0.95)	0:12
Syntax	34.7 (0.99)	32.3 (0.95)	0:45
Hier.($\lambda^\star$)	-	32.1 (0.95)	0:12
Preference Grammar: $\beta_R = 100$			
$\beta_P = 1$	-	32.5 (0.96)	3:00
$\beta_P = 2$	34.6 (0.99)	32.6 (0.95)	3:00
$\beta_P = 5$	-	32.5 (0.95)	3:20
Preference Grammar: $\beta_R = 10$			
$\beta_P = 1$	-	32.5 (0.95)	1:03
$\beta_P = 2$	-	32.6 (0.95)	1:10
$\beta_P = 5$	-	32.5 (0.95)	1:10

Table 3: Translation quality and test set translation time (using 50 machines with 2 tasks per machine) measured by the BLEU metric for the NIST task. NIST 2006 is used as the development (Dev.) corpus and NIST 2007 is used as the unseen evaluation corpus (Test). Dev. scores are reported for systems that have been separately MERT trained, Pref. systems share parameters from a single MERT training. Systems are described in the text.

7 Related Work

There have been significant efforts in the both the monolingual parsing and machine translation literature to address the impact of the MAP approximation and the choice of labels in their respective models; we survey the work most closely related to our approach. May and Knight (2006) extract n-best lists containing unique translations rather than unique derivations, while Kumar and Byrne (2004) use the Minimum Bayes Risk decision rule to select the lowest risk (highest BLEU score) translation rather than derivation from an n-best list. Tromble et al. (2008) extend this work to lattice structures. All of these approaches only marginalize over alternative candidate derivations generated by a MAP-driven decoding process. More recently, work by Blunsom et al. (2007) propose a purely discriminative model whose decoding step approximates the selection of the most likely translation via beam search. Matsusaki et al. (2005) and Petrov et al. (2006) propose automatically learning annotations that add information to categories to improve monolingual parsing quality. Since the parsing task requires selecting the most non-annotated tree, the annotations add an additional level of structure that must be marginalized during search. They demonstrate improvements in parse quality only when a variational approximation is used to select the most likely *unannotated* tree rather than simply stripping annotations from the MAP annotated tree. In our work, we focused on approximating the selection of the most likely unlabeled derivation during search, rather than as a post-processing operation; the methods described above might improve this approximation, at some computational expense.

8 Conclusions and Future Work

We have proposed a novel grammar formalism that replaces hard syntactic constraints with "soft" preferences. These preferences are used to compute a machine translation feature ($p_{\text{syn}}(d)$) that scores unlabeled derivations, taking into account traditional syntactic constraints. Representing syntactic constraints as a feature allows MERT to train the corresponding weight for this feature relative to others in the model, allowing systems to learn the relative importance of labels for particular resource and language scenarios as well as for alternative approaches to labeling PSCFG rules.

This approach takes a step toward addressing the fragmentation problems of decoding based on maximum-weighted derivations, by summing the contributions of compatible label configurations rather than forcing them to compete. We have suggested an efficient technique to approximate $p_{\text{syn}}(d)$ that takes advantage of a natural factoring of derivation scores. Our approach results in improvements in translation quality on small and medium resource translation tasks. In future work we plan to focus on methods to improve on the integration of the $p_{\text{syn}}(d)$ feature during decoding and techniques that allow us consider more of the search space through less pruning.

Acknowledgements

We appreciate helpful comments from three anonymous reviewers. Venugopal and Zollmann were supported by a Google Research Award. Smith was supported by NSF grant IIS-0836431.

References

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2007. A discriminative latent variable model for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Francisco Casacuberta and Colin de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In *Proc. of the 5th International Colloquium on Grammatical Inference: Algorithms and Applications*.

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

David Chiang. 2007. Hierarchical phrase based translation. *Computational Linguistics*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inferences and training of context-rich syntax translation models. In *Proceedings of the Annual Meeting of the Association for Compuational Linguistics (ACL)*, Sydney, Australia.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the Annual Meeting of the Association for Compuational Linguistics (ACL)*.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics, Squibs and Discussion*.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, Boston,MA, May 27-June 1.

Takuya Matsusaki, Yusuke Miyao, and Junichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jonathan May and Kevin Knight. 2006. A better N-best list: Practical determinization of weighted finite tree automata. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Compuational Linguistics (ACL)*.

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

Michael Paul. 2006. Overview of the IWSLT 2006 evaluation campaign. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the Annual Meeting of the Association for Compuational Linguistics (ACL)*.

Khalil Sima'an. 2002. Computational complexity of probabilistic disambiguation. *Grammars*, 5(2):125–151.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ashish Venugopal, Andreas Zollmann, and Stephan Vogel. 2007. An efficient two-pass approach to Synchronous-CFG driven statistical MT. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Conference (HLT/NAACL)*.

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

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

Using a Dependency Parser to Improve SMT for Subject-Object-Verb Languages

Peng Xu, Jaeho Kang, Michael Ringgaard and Franz Och
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043, USA
{xp,jhkang,ringgaard,och}@google.com

Abstract

We introduce a novel precedence reordering approach based on a dependency parser to statistical machine translation systems. Similar to other preprocessing reordering approaches, our method can efficiently incorporate linguistic knowledge into SMT systems without increasing the complexity of decoding. For a set of five subject-object-verb (SOV) order languages, we show significant improvements in BLEU scores when translating from English, compared to other reordering approaches, in state-of-the-art phrase-based SMT systems.

1 Introduction

Over the past ten years, statistical machine translation has seen many exciting developments. Phrase-based systems (Och, 2002; Koehn et.al., 2003; Och and Ney, 2004) advanced the machine translation field by allowing translations of word sequences (a.k.a., phrases) instead of single words. This approach has since been the state-of-the-art because of its robustness in modeling local word reordering and the existence of an efficient dynamic programming decoding algorithm.

However, when phrase-based systems are used between languages with very different word orders, such as between subject-verb-object (SVO) and subject-object-verb (SOV) languages, long distance reordering becomes one of the key weaknesses. Many reordering methods have been proposed in recent years to address this problem in different aspects.

The first class of approaches tries to explicitly model phrase reordering distances. Distance based distortion model (Och, 2002; Koehn et.al., 2003) is a simple way of modeling phrase level reordering. It penalizes non-monotonicity by applying a weight to the number of words between two source phrases corresponding to two consecutive target phrases. Later on, this model was extended to lexicalized phrase reordering (Tillmann, 2004; Koehn, et.al., 2005; Al-Onaizan and Papineni, 2006) by applying different weights to different phrases. Most recently, a hierarchical phrase reordering model (Galley and Manning, 2008) was proposed to dynamically determine phrase boundaries using efficient shift-reduce parsing. Along this line of research, discriminative reordering models based on a maximum entropy classifier (Zens and Ney, 2006; Xiong, et.al., 2006) also showed improvements over the distance based distortion model. None of these reordering models changes the word alignment step in SMT systems, therefore, they can not recover from the word alignment errors. These models are also limited by a maximum allowed reordering distance often used in decoding.

The second class of approaches puts syntactic analysis of the target language into both modeling and decoding. It has been shown that direct modeling of target language constituents movement in either constituency trees (Yamada and Knight, 2001; Galley et.al., 2006; Zollmann et.al., 2008) or dependency trees (Quirk, et.al., 2005) can result in significant improvements in translation quality for translating languages like Chinese and Arabic into English. A simpler alternative, the hierarchical phrase-based

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 245–253,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

approach (Chiang, 2005; Wu, 1997) also showed promising results for translating Chinese to English. Similar to the distance based reordering models, the syntactical or hierarchical approaches also rely on other models to get word alignments. These models typically combine machine translation decoding with chart parsing, therefore significantly increase the decoding complexity. Even though some recent work has shown great improvements in decoding efficiency for syntactical and hierarchical approaches (Huang and Chiang, 2007), they are still not as efficient as phrase-based systems, especially when higher order language models are used.

Finally, researchers have also tried to put source language syntax into reordering in machine translation. Syntactical analysis of source language can be used to deterministically reorder input sentences (Xia and McCord, 2004; Collins et.al., 2005; Wang et.al., 2007; Habash, 2007), or to provide multiple orderings as weighted options (Zhang et.al., 2007; Li et.al., 2007; Elming, 2008). In these approaches, input source sentences are reordered based on syntactic analysis and some reordering rules at preprocessing step. The reordering rules can be either manually written or automatically extracted from data. Deterministic reordering based on syntactic analysis for the input sentences provides a good way of resolving long distance reordering, without introducing complexity to the decoding process. Therefore, it can be efficiently incorporated into phrase-based systems. Furthermore, when the same preprocessing reordering is performed for the training data, we can still apply other reordering approaches, such as distance based reordering and hierarchical phrase reordering, to capture additional local reordering phenomena that are not captured by the preprocessing reordering. The work presented in this paper is largely motivated by the preprocessing reordering approaches.

In the rest of the paper, we first introduce our dependency parser based reordering approach based on the analysis of the key issues when translating SVO languages to SOV languages. Then, we show experimental results of applying this approach to phrase-based SMT systems for translating from English to five SOV languages (Korean, Japanese, Hindi, Urdu and Turkish). After showing that this approach can also be beneficial for hierarchical phrase-based systems, we will conclude the paper with future research directions.

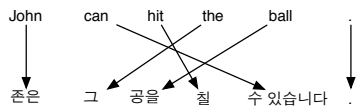


Figure 1: Example Alignment Between an English and a Korean Sentence

2 Translation between SVO and SOV Languages

In linguistics, it is possible to define a basic word order in terms of the verb (V) and its arguments, subject (S) and object (O). Among all six possible permutations, SVO and SOV are the most common. Therefore, translating between SVO and SOV languages is a very important area to study. We use English as a representative of SVO languages and Korean as a representative for SOV languages in our discussion about the word orders.

Figure 1 gives an example sentence in English and its corresponding translation in Korean, along with the alignments between the words. Assume that we split the sentences into four phrases: (John , 존은), (can hit , 칠 수 있습니다), (the ball , 그 공을) and (. , .). Since a phrase-based decoder generates the translation from left to right, the following steps need to happen when we translate from English to Korean:

- Starts from the beginning of the sentence, translates “John” to “존은”;
- Jumps to the right by two words, translates “the ball” to “그 공을”;
- Jumps to the left by four words, translates “can hit” to “칠 수 있습니다”;
- Finally, jumps to the right by two words, translates “.” to “.”.

It is clear that in order for the phrase-based decoder to successfully carry out all of the reordering steps, a very strong reordering model is required. When the sentence gets longer with more complex structure, the number of words to move over during decoding can be quite high. Imagine when we translate

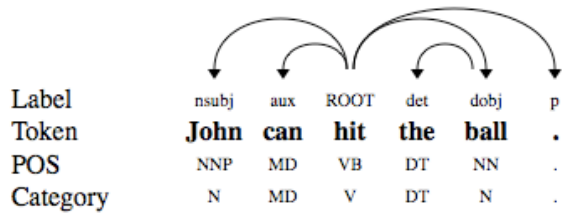


Figure 2: Dependency Parse Tree of an Example English Sentence

the sentence "**English is used as the first or second language in many countries around the world .**". The decoder needs to make a jump of 13 words in order to put the translation of "**is used**" at the end of the translation. Normally in a phrase-based decoder, very long distance reordering is not allowed because of efficiency considerations. Therefore, it is very difficult in general to translate English into Korean with proper word order.

However, knowing the dependency parse trees of the English sentences may simplify the reordering problem significantly. In the simple example in Figure 1, if we analyze the English sentence and know that "John" is the subject, "can hit" is the verb and "the ball" is the object, we can reorder the English into SOV order. The resulting sentence "John the ball can hit ." will only need monotonic translation. This motivates us to use a dependency parser for English to perform the reordering.

3 Precedence Reordering Based on a Dependency Parser

Figure 2 shows the dependency tree for the example sentence in the previous section. In this parse, the verb "hit" has four children: a subject noun "John", an auxiliary verb "can", an object noun "ball" and a punctuation ".". When transforming the sentence to SOV order, we need to move the object noun and the subtree rooted at it to the front of the head verb, but after the subject noun. We can have a simple rule to achieve this.

However, in reality, there are many possible children for a verb. These children have some relative ordering that is typically fixed for SOV languages. In order to describe this kind of ordering, we propose precedence reordering rules based on a dependency parse tree. All rules here are based English and Korean examples, but they also apply to other SOV languages, as we will show later empirically.

A precedence reordering rule is a mapping from T to a set of tuples $\{(L, W, O)\}$, where T is the part-of-speech (POS) tag of the head in a dependency parse tree node, L is a dependency label for a child node, W is a weight indicating the order of that child node and O is the type of order (either *NORMAL* or *REVERSE*). The type of order is only used when we have multiple children with the same weight, while the weight is used to determine the relative order of the children, going from largest to smallest. The weight can be any real valued number. The order type NORMAL means we preserve the original order of the children, while REVERSE means we flip the order. We reserve a special label $self$ to refer to the head node itself so that we can apply a weight to the head, too. We will call this tuple a *precedence tuple* in later discussions. In this study, we use manually created rules only.

Suppose we have a precedence rule: VB $\rightarrow$ (nsubj, 2, NORMAL), (dobj, 1, NORMAL), (self, 0, NORMAL). For the example shown in Figure 2, we would apply it to the ROOT node and result in "John the ball can hit .".

Given a set of rules, we apply them in a dependency tree recursively starting from the root node. If the POS tag of a node matches the left-hand-side of a rule, the rule is applied and the order of the sentence is changed. We go through all children of the node and get the precedence weights for them from the set of precedence tuples. If we encounter a child node that has a dependency label not listed in the set of tuples, we give it a default weight of 0 and default order type of NORMAL. The children nodes are sorted according to their weights from highest to lowest, and nodes with the same weights are ordered according to the type of order defined in the rule.

3.1 Verb Precedence Rules

Verb movement is the most important movement when translating from English (SVO) to Korean (SOV). In a dependency parse tree, a verb node can potentially have many children. For example, auxiliary and passive auxiliary verbs are often grouped together with the main verb and moved together with it. The order, however, is reversed after the movement. In the example of Figure 2, the correct Korean

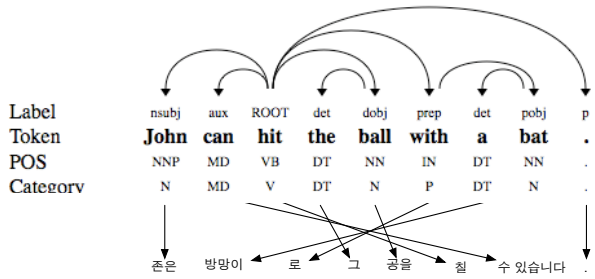


Figure 3: Dependency Parse Tree with Alignment for a Sentence with Preposition Modifier

word order is "**칠 (hit) 수 있습니다(can)** . Other categories that are in the same group are phrasal verb particle and negation.

If the verb in an English sentence has a prepositional phrase as a child, the prepositional phrase is often placed before the direct object in the Korean counterpart. As shown in Figure 3, "방망이 로" ("with a bat") is actually between "존은" ("John") and "그 공을" ("the ball").

Another common reordering phenomenon is when a verb has an adverbial clause modifier. In that case, the whole adverbial clause is moved together to be in front of the subject of the main sentence. Inside the adverbial clause, the ordering follows the same verb reordering rules, so we recursively reorder the clause.

Our verb precedence rule, as in Table 1, can cover all of the above reordering phenomena. One way to interpret this rule set is as follows: for any node whose POS tag is matches VB* (VB, VBZ, VBD, VBP, VBN, VBG), we group the children node that are phrasal verb particle (prt), auxiliary verb (aux), passive auxiliary verb (auxpass), negation (neg) and the verb itself (self) together and reverse them. This verb group is moved to the end of the sentence. We move adverbial clause modifier to the beginning of the sentence, followed by a group of noun subject (nsubj), preposition modifier and anything else not listed in the table, in their original order. Right before the verb group, we put the direct object (dobj). Note that all of the children are optional.

3.2 Adjective Precedence Rules

Similar to the verbs, adjectives can also take an auxiliary verb, a passive auxiliary verb and a negation

T	(L, W, O)
VB*	(advcl, 1, NORMAL) (nsubj, 0, NORMAL) (prep, 0, NORMAL) (dobj, -1, NORMAL) (prt, -2, REVERSE) (aux, -2, REVERSE) (auxpass, -2, REVERSE) (neg, -2, REVERSE) (self, -2, REVERSE)
JJ or JJS or JJR	(advcl, 1, NORMAL) (self, -1, NORMAL) (aux, -2, REVERSE) (auxpass, -2, REVERSE) (neg, -2, REVERSE) (cop, -2, REVERSE)
NN or NNS	(prep, 2, NORMAL) (rcmod, 1, NORMAL) (self, 0, NORMAL)
IN or TO	(pobj, 1, NORMAL) (self, -1, NORMAL)

Table 1: Precedence Rules to Reorder English to SOV Language Order (These rules were extracted manually by a bilingual speaker after looking at some text book examples in English and Korean, and the dependency parse trees of the English examples.)

as modifiers. In such cases, the change in order from English to Korean is similar to the verb rule, except that the head adjective itself should be in front of the verbs. Therefore, in our adjective precedence rule in the second panel of Table 1, we group the auxiliary verb, the passive auxiliary verb and the negation and move them together after reversing their order. They are moved to right after the head adjective, which is put after any other modifiers.

For both verb and adjective precedence rules, we also apply some heuristics to prevent excessive movements. In order to do this, we disallow any movement across punctuation and conjunctions. Therefore, for sentences like "**John hit the ball but Sam threw the ball**", the reordering result would be "**John the ball hit but Sam the ball threw**", instead of "**John the ball but Sam the ball threw hit**".

3.3 Noun and Preposition Precedence Rules

In Korean, when a noun is modified by a prepositional phrase, such as in "**the way to happiness**", the prepositional phrase is usually moved in front of the noun, resulting in "**행복 (happiness) 으로 가는 길 (to the way)**" . Similarly for relative clause modifier, it is also reordered to the front of the head noun. For preposition head node with an object modifier,

the order is the object first and the preposition last. One example is "**with a bat**" in Figure 3. It corresponds to "**방망이 (a bat) 로(with)**". We handle these types of reordering by the noun and preposition precedence rules in the third and fourth panel of Table 1.

With the rules defined in Table 1, we now show a more complex example in Figure 4. First, the ROOT node matches an adjective rule, with four children nodes labeled as (csubj, cop, advcl, p), and with precedence weights of (0, -2, 1, 0). The ROOT node itself has a weight of -1. After reordering, the sentence becomes: "**because we do n't know what the future has Living exciting is .**". Note that the whole adverbial phrase rooted at "**know**" is moved to the beginning of the sentence. After that, we see that the child node rooted at "**know**" matches a verb rule, with five children nodes labeled as (mark, nsubj, aux, neg, ccomp), with weights (0, 0, -2, -2, 0). In this case, the verb itself also has weight -2. Now we have two groups of nodes, with weight 0 and -2, respectively. The first group has a NORMAL order and the second group has a REVERSE order. After reordering, the sentence becomes: "**because we what the future has know n't do Living exciting is .**". Finally, we have another node rooted at "**has**" that matches the verb rule again. After the final reordering, we end up with the sentence: "**because we the future what has know n't do Living exciting is .**". We can see in Figure 4 that this sentence has an almost monotonic alignment with a reasonable Korean translation shown in the figure[1].

4 Related Work

As we mentioned in our introduction, there have been several studies in applying source sentence reordering using syntactical analysis for statistical machine translation. Our precedence reordering approach based on a dependency parser is motivated by those previous works, but we also distinguish from their studies in various ways.

Several approaches use syntactical analysis to provide multiple source sentence reordering options through word lattices (Zhang et.al., 2007; Li et.al., 2007; Elming, 2008). A key difference between their approaches and ours is that they do not perform reordering during training. Therefore, they would need to rely on reorder units that are likely not violating "phrase" boundaries. However, since we reorder both training and test data, our system operates in a matched condition. They also focus on either Chinese to English (Zhang et.al., 2007; Li et.al., 2007) or English to Danish (Elming, 2008), which arguably have less long distance reordering than between English and SOV languages.

Studies most similar to ours are those preprocessing reordering approaches (Xia and McCord, 2004; Collins et.al., 2005; Wang et.al., 2007; Habash, 2007). They all perform reordering during preprocessing based on either automatically extracted syntactic rules (Xia and McCord, 2004; Habash, 2007) or manually written rules (Collins et.al., 2005; Wang et.al., 2007). Compared to these approaches, our work has a few differences. First of all, we study a wide range of SOV languages using manually extracted precedence rules, not just for one language like in these studies. Second, as we will show in the next section, we compare our approach to a very strong baseline with more advanced distance based reordering model, not just the simplest distortion model. Third, our precedence reordering rules, like those in Habash, 2007, are more flexible than those other rules. Using just one verb rule, we can perform the reordering of subject, object, preposition modifier, auxiliary verb, negation and the head verb. Although we use manually written rules in this study, it is possible to learn our rules automatically from alignments, similarly to Habash, 2007. However, unlike Habash, 2007, our manually written rules handle unseen children and their order naturally because we have a default precedence weight and order type, and we do not need to match an often too specific condition, but rather just treat all children independently. Therefore, we do not need to use any backoff scheme in order to have a broad coverage. Fourth, we use dependency parse trees rather than constituency trees.

There has been some work on syntactic word order model for English to Japanese machine translation (Chang and Toutanova, 2007). In this work, a global word order model is proposed based on features including word bigram of the target sentence, displacements and POS tags on both source and tar-

[1] We could have improved the rules by using a weight of -3 for the label "*mark*", but it was not in our original set of rules.

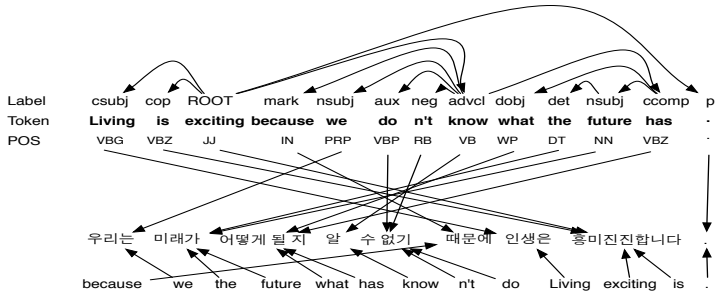


Figure 4: A Complex Reordering Example (Reordered English sentence and alignments are at the bottom.)

get sides. They build a log-linear model using these features and apply the model to re-rank N-best lists from a baseline decoder. Although we also study the reordering problem in English to Japanese translation, our approach is to incorporate the linguistically motivated reordering directly into modeling and decoding.

System	Source	Target
English→Korean	303M	267M
English→Japanese	316M	350M
English→Hindi	16M	17M
English→Urdu	17M	19M
English→Turkish	83M	76M

Table 2: Training Corpus Statistics (#words) of Systems for 5 SOV Languages

5 Experiments

We carried out all our experiments based on a state-of-the-art phrase-based statistical machine translation system. When training a system for English to any of the 5 SOV languages, the word alignment step includes 3 iterations of IBM Model-1 training and 2 iterations of HMM training. We do not use Model-4 because it is slow and it does not add much value to our systems in a pilot study. We use the standard phrase extraction algorithm (Koehn et.al., 2003) to get all phrases up to length 5. In addition to the regular distance distortion model, we incorporate a maximum entropy based lexicalized phrase reordering model (Zens and Ney, 2006) as a feature used in decoding. In this model, we use 4 reordering classes ($+1$, > 1, -1, < -1) and words from both source and target as features. For source words, we use the current aligned word, the word before the current aligned word and the next aligned word; for target words, we use the previous two words in the immediate history. Using this type of features makes it possible to directly use the maximum entropy model in the decoding process (Zens and Ney, 2006). The maximum entropy models are trained on all events extracted from training data word alignments using the LBFGS algorithm (Malouf, 2002). Overall for decoding, we use between 20 to 30 features, whose weights are optimized using MERT (Och, 2003), with an implementation based on the lattice MERT (Macherey et.al., 2008).

For parallel training data, we use an in-house collection of parallel documents. They come from various sources with a substantial portion coming from the web after using simple heuristics to identify potential document pairs. Therefore, for some documents in the training data, we do not necessarily have the exact clean translations. Table 2 shows the actual statistics about the training data for all five languages we study. For all 5 SOV languages, we use the target side of the parallel data and some more monolingual text from crawling the web to build 4-gram language models.

We also collected about 10K English sentences from the web randomly. Among them, 9.5K are used as evaluation data. Those sentences were translated by humans to all 5 SOV languages studied in this paper. Each sentence has only one reference translation. We split them into 3 subsets: *dev* contains 3,500 sentences, *test* contains 1,000 sentences and the rest of 5,000 sentences are used in a *blindtest* set. The dev set is used to perform MERT training, while the test set is used to select trained weights due to some nondeterminism of MERT training. We use IBM BLEU (Papineni et al., 2002) to evaluate

our translations and use character level BLEU for Korean and Japanese.

5.1 Preprocessing Reordering and Reordering Models

We first compare our precedence rules based preprocessing reordering with the maximum entropy based lexicalized reordering model. In Table 3, *Baseline* is our system with both a distance distortion model and the maximum entropy based lexicalized reordering model. For all results reported in this section, we used a maximum allowed reordering distance of 10. In order to see how the lexicalized reordering model performs, we also included systems with and without it (*-LR* means without it). *PR* is our proposed approach in this paper. Note that since we apply precedence reordering rules during preprocessing, we can combine this approach with any other reordering models used during decoding. The only difference is that with the precedence reordering, we would have a different phrase table and in the case of *LR*, different maximum entropy models.

In order to implement the precedence rules, we need a dependency parser. We choose to use a deterministic inductive dependency parser (Nivre and Scholz, 2004) for its efficiency and good accuracy. Our implementation of the deterministic dependency parser using maximum entropy models as the underlying classifiers achieves 87.8% labeled attachment score and 88.8% unlabeled attachment score on standard Penn Treebank evaluation.

As our results in Table 3 show, for all 5 languages, by using the precedence reordering rules as described in Table 1, we achieve significantly better BLEU scores compared to the baseline system. In the table, We use two stars ($**$) to mean that the statistical significance test using the bootstrap method (Koehn, 2004) gives an above 95% significance level when compared to the baselie. We measured the statistical significance level only for the blindtest data.

Note that for Korean and Japanese, our precedence reordering rules achieve better absolute BLEU score improvements than for Hindi, Urdu and Turkish. Since we only analyzed English and Korean sentences, it is possible that our rules are more geared toward Korean. Japanese has almost exactly the same word order as Korean, so we could assume the benefits can carry over to Japanese.

Language	System	dev	test	blind
Korean	BL	25.8	27.0	26.2
	-LR	24.7	25.6	25.1
	-LR+PR	27.3	28.3	27.5**
	+PR	27.8	28.7	27.9**
Japanese	BL	29.5	29.3	29.3
	-LR	29.2	29.0	29.0
	-LR+PR	30.3	31.0	30.6**
	+PR	30.7	31.2	31.1**
Hindi	BL	19.1	18.9	18.3
	-LR	17.4	17.1	16.4
	-LR+PR	19.6	18.8	18.7**
	+PR	19.9	18.9	18.8**
Urdu	BL	9.7	9.5	8.9
	-LR	9.1	8.6	8.2
	-LR+PR	10.0	9.6	9.6**
	+PR	10.0	9.8	9.6**
Turkish	BL	10.0	10.5	9.8
	-LR	9.1	10.0	9.0
	-LR+PR	10.5	11.0	10.3**
	+PR	10.5	10.9	10.4**

Table 3: BLEU Scores on Dev, Test and Blindtest for English to 5 SOV Languages with Various Reordering Options (BL means baseline, LR means maximum entropy based lexialized phrase reordering model, PR means precedence rules based preprocessing reordering.)

5.2 Reordering Constraints

One of our motivations of using the precedence reordering rules is that English will look like SOV languages in word order after reordering. Therefore, even monotone decoding should be able to produce better translations. To see this, we carried out a controlled experiment, using Korean as an example.

Clearly, after applying the precedence reordering rules, our English to Korean system is not sensitive to the maximum allowed reordering distance anymore. As shown in Figure 5, without the rules, the blindtest BLEU scores improve monotonically as the allowed reordering distance increases. This indicates that the order difference between English and Korean is very significant. Since smaller allowed reordering distance directly corresponds to decoding time, we can see that with the same decoding speed, our proposed approach can achieve almost 5% BLEU score improvements on blindtest set.

5.3 Preprocessing Reordering and Hierarchical Model

The hierarchical phrase-based approach has been successfully applied to several systems (Chiang,

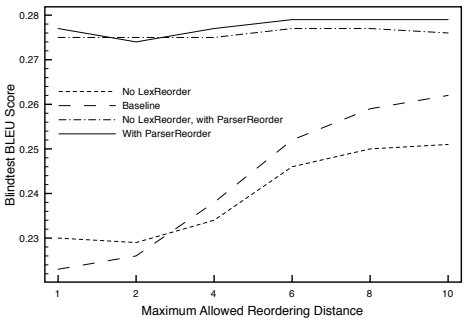


Figure 5: Blindtest BLEU Score for Different Maximum Allowed Reordering Distance for English to Korean Systems with Different Reordering Options

Language	System	dev	test	blind
Korean	PR	27.8	28.7	27.9
	Hier	27.4	27.7	27.9
	PR+Hier	28.5	29.1	28.8**
Japanese	PR	30.7	31.2	31.1**
	Hier	30.5	30.6	30.5
	PR+Hier	31.0	31.3	31.1**
Hindi	PR	19.9	18.9	18.8
	Hier	20.3	20.3	19.3
	PR+Hier	20.0	19.7	19.3
Urdu	PR	10.0	9.8	9.6
	Hier	10.4	10.3	10.0
	PR+Hier	11.2	10.7	10.7**
Turkish	PR	10.5	10.9	10.4
	Hier	11.0	11.8	10.5
	PR+Hier	11.1	11.6	10.9**

Table 4: BLEU Scores on Dev, Test and Blindtest for English to 5 SOV Languages in Hierarchical Phrase-based Systems (PR is precedence rules based preprocessing reordering, same as in Table 3, while Hier is the hierarchical system.)

2005; Zollmann et.al., 2008). Since hierarchical phrase-based systems can capture long distance reordering by using a PSCFG model, we expect it to perform well in English to SOV language systems.

We use the same training data as described in the previous sections for building hierarchical systems. The same 4-gram language models are also used for the 5 SOV languages. We adopt the SAMT package (Zollmann and Venugopal, 2006) and follow similar settings as Zollmann et.al., 2008. We allow each rule to have at most 6 items on the source side, including nonterminals and extract rules from initial phrases of maximum length 12. During decoding, we allow application of all rules of the grammar for chart items spanning up to 12 source words.

Since our precedence reordering applies at preprocessing step, we can train a hierarchical system after applying the reordering rules. When doing so, we use exactly the same settings as a regular hierarchical system. The results for both hierarchical systems and those combined with the precedence reordering are shown in Table 4, together with the best normal phrase-based systems we copy from Table 3. Here again, we mark any blindtest BLEU score that is better than the corresponding hierarchical system with confidence level above 95%. Note that the hierarchical systems can not use the maximum entropy based lexicalized phrase reordering models.

Except for Hindi, applying the precedence reordering rules in a hierarchical system can achieve statistically significant improvements over a normal hierarchical system. We conjecture that this may be because of the simplicity of our reordering rules. Other than the reordering phenomena covered by our rules in Table 1, there could be still some local or long distance reordering. Therefore, using a hierarchical phrase-based system can improve those cases. Another possible reason is that after the reordering rules apply in preprocessing, English sentences in the training data are very close to the SOV order. As a result, EM training becomes much easier and word alignment quality becomes better. Therefore, a hierarchical phrase-based system can extract better rules and hence achievesbetter translation quality.

We also point out that hierarchical phrase-based systems require a chart parsing algorithm during decoding. Compared to the efficient dynamic programming in phrase-based systems, it is much slower. This makes our approach more appealing in a realtime statistical machine translation system.

6 Conclusion

In this paper, we present a novel precedence reordering approach based on a dependency parser. We successfully applied this approach to systems translating English to 5 SOV languages: Korean, Japanese, Hindi, Urdu and Turkish. For all 5 languages, we achieve statistically significant improvements in BLEU scores over a state-of-the-art phrase-based baseline system. The amount of training data for the 5 languages varies from around 17M to more than 350M words, including some noisy data from

the web. Our proposed approach has shown to be robust and versatile. For 4 out of the 5 languages, our approach can even significantly improve over a hierarchical phrase-based baseline system. As far as we know, we are the first to show that such reordering rules benefit several SOV languages.

We believe our rules are flexible and can cover many linguistic reordering phenomena. The format of our rules also makes it possible to automatically extract rules from word aligned corpora. In the future, we plan to investigate along this direction and extend the rules to languages other than SOV.

The preprocessing reordering like ours is known to be sensitive to parser errors. Some preliminary error analysis already show that indeed some sentences suffer from parser errors. In the recent years, several studies have tried to address this issue by using a word lattice instead of one reordering as input (Zhang et.al., 2007; Li et.al., 2007; Elming, 2008). Although there is clearly room for improvements, we also feel that using one reordering during training may not be good enough either. It would be very interesting to investigate ways to have efficient procedure for training EM models and getting word alignments using word lattices on the source side of the parallel data. Along this line of research, we think some kind of tree-to-string model (Liu et.al., 2006) could be interesting directions to pursue.

References

Yaser Al-Onaizan and Kishore Papineni 2006. Distortion Models for Statistical Machine Translation *In Proceedings of ACL*

Pi-Chuan Chang and Kristina Toutanova 2007. A Discriminative Syntactic Word Order Model for Machine Translation *In Proceedings of ACL*

David Chiang 2005. A Hierarchical Phrase-based Model for Statistical Machine Translation *In Proceedings of ACL*

Michael Collins, Philipp Koehn and Ivona Kucerova 2005. Clause Restructuring for Statistical Machine Translation *In Proceedings of ACL*

Jakob Elming 2008. Syntactic Reordering Integrated with Phrase-based SMT *In Proceedings of COLING*

Michel Galley and Christopher D. Manning 2008. A Simple and Effective Hierarchical Phrase Reordering Model *In Proceedings of EMNLP*

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang and Ignacio Thayer 2006. Scalable Inference and Training of Context-Rich Syntactic Translation Models *In Proceedings of COLING-ACL*

Nizar Habash 2007. Syntactic Preprocessing for Statistical Machine Translation *In Proceedings of 11th MT Summit*

Liang Huang and David Chiang 2007. Forest Rescoring: Faster Decoding with Integrated Language Models, *In Proceedings of ACL*

Philipp Koehn 2004. Statistical Significance Tests for Machine Translation Evaluation *In Proceedings of EMNLP*

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne and David Talbot 2005. Edinborgh System Description for the 2005 IWSLT Speech Translation Evaluation *In International Workshop on Spoken Language Translation*

Philipp Koehn, Franz J. Och and Daniel Marcu 2003. Statistical Phrase-based Translation, *In Proceedings of HLT-NAACL*

Chi-Ho Li, Dongdong Zhang, Mu Li, Ming Zhou, Minghui Li and Yi Guan 2007. A Probabilistic Approach to Syntax-based Reordering for Statistical Machine Translation, *In Proceedings of ACL*

Yang Liu, Qun Liu and Shouxun Lin 2006. Tree-to-string Alignment Template for Statistical Machine Translation, *In Proceedings of COLING-ACL*

Wolfgang Macherey, Franz J. Och, Ignacio Thayer and Jakob Uszkoreit 2008. Lattice-based Minimum Error Rate Training for Statistical Machine Translation *In Proceedings of EMNLP*

Robert Malouf 2002. A comparison of algorithms for maximum entropy parameter estimation *In Proceedings of the Sixth Workshop on Computational Language Learning (CoNLL-2002)*

Joakim Nivre and Mario Scholz 2004. Deterministic Dependency Parsing for English Text. *In Proceedings of COLING*

Franz J. Och 2002. Statistical Machine Translation: From Single Word Models to Alignment Template Ph.D. Thesis, RWTH Aachen, Germany

Franz J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. *In Proceedings of ACL*

Franz J. Och and Hermann Ney 2004. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics, 30:417-449*

Kishore Papineni, Roukos, Salim et al. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. *In Proceedings of ACL*

Chris Quirk, Arul Menezes and Colin Cherry 2005. Dependency Tree Translation: Syntactically Informed Phrasal SMT *In Proceedings of ACL*

Christoph Tillmann 2004. A Block Orientation Model for Statistical Machine Translation *In Proceedings of HLT-NAACL*

Chao Wang, Michael Collins and Philipp Koehn 2007. Chinese Syntactic Reordering for Statistical Machine Translation *In Proceedings of EMNLP-CoNLL*

Dekai Wu 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpus *In 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*

Deyi Xiong, Qun Liu and Shouxun Lin 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation *In Proceedings of COLING-ACL*

Kenji Yamada and Kevin Knight 2001. A Syntax-based Statistical Translation Model *In Proceedings of ACL*

Yuqi Zhang, Richard Zens and Hermann Ney 2007. Improve Chunk-level Reordering for Statistical Machine Translation *In Proceedings of IWSLT*

Richard Zens and Hermann Ney 2006. Discriminative Reordering Models for Statistical Machine Translation *In Proceedings of the Workshop on Statistical Machine Translation, HLT-NAACL* pages 55-63

Andreas Zollmann and Ashish Venugopal 2006. Syntax Augmented Machine Translation via Chart Parsing *In Proceedings of NAACL 2006 - Workshop on Statistical Machine Translation*

Andreas Zollmann, Ashish Venugopal, Franz Och and Jay Ponte 2008. A Systematic Comparison of Phrase-Based, Hierarchical and Syntax-Augmented Statistical MT *In Proceedings of COLING*

Learning Bilingual Linguistic Reordering Model for Statistical Machine Translation

Han-Bin Chen, Jian-Cheng Wu and Jason S. Chang
Department of Computer Science
National Tsing Hua University
101, Guangfu Road, Hsinchu, Taiwan
{hanbin,d928322,jschang}@cs.nthu.edu.tw

Abstract

In this paper, we propose a method for learning reordering model for BTG-based statistical machine translation (SMT). The model focuses on linguistic features from bilingual phrases. Our method involves extracting reordering examples as well as features such as part-of-speech and word class from aligned parallel sentences. The features are classified with special considerations of phrase lengths. We then use these features to train the maximum entropy (ME) reordering model. With the model, we performed Chinese-to-English translation tasks. Experimental results show that our bilingual linguistic model outperforms the state-of-the-art phrase-based and BTG-based SMT systems by improvements of 2.41 and 1.31 BLEU points respectively.

1 Introduction

Bracketing Transduction Grammar (BTG) is a special case of Synchronous Context Free Grammar (SCFG), with binary branching rules that are either straight or inverted. BTG is widely adopted in SMT systems, because of its good trade-off between efficiency and expressiveness (Wu, 1996). In BTG, the ratio of legal alignments and all possible alignment in a translation pair drops drastically especially for long sentences, yet it still covers most of the syntactic diversities between two languages.

It is common to utilize phrase translation in BTG systems. For example in (Xiong et al., 2006), source sentences are segmented into phrases. Each sequences of consecutive phrases, mapping to cells in a CKY matrix, are then translated through a bilingual phrase table and scored as implemented in (Koehn et al., 2005; Chiang, 2005). In other words, their system shares the same phrase table with standard phrase-based SMT systems.

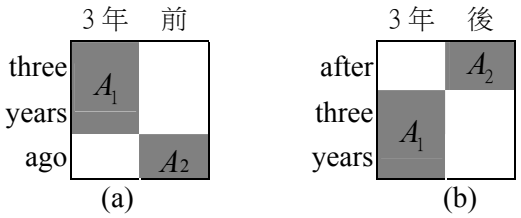


Figure 1: Two reordering examples, with straight rule applied in (a), and inverted rule in (b).

On the other hand, there are various proposed BTG reordering models to predict correct orientations between neighboring blocks (bilingual phrases). In Figure 1, for example, the role of reordering model is to predict correct orientations of neighboring blocks A_1 and A_2. In flat model (Wu, 1996; Zens et al., 2004; Kumar and Byrne, 2005), reordering probabilities are assigned uniformly during decoding, and can be tuned depending on different language pairs. It is clear, however, that this kind of model would suffer when the dominant rule is wrongly applied.

Predicting orientations in BTG depending on context information can be achieved with lexical features. For example, Xiong et al. (2006) proposed MEBTG, based on maximum entropy (ME) classification with words as features. In MEBTG, first words of blocks are considered as the features, which are then used to train a ME model

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 254–262,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

for predicting orientations of neighboring blocks. Xiong et al. (2008b) proposed a linguistically annotated BTG (LABTG), in which linguistic features such as POS and syntactic labels from source-side parse trees are used. Both MEBTG and LABTG achieved significant improvements over phrase-based Pharaoh (Koehn, 2004) and Moses (Koehn et al., 2007) respectively, on Chinese-to-English translation tasks.

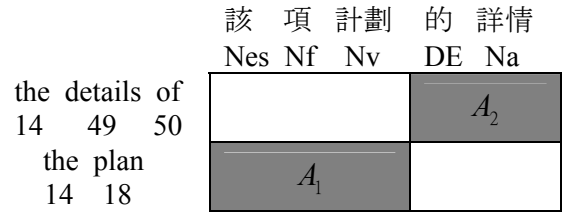


Figure 2: An inversion reordering example, with POS below source words, and class numbers below target words.

However, current BTG-based reordering methods have been limited by the features used. Information might not be sufficient or representative, if only the first (or tail) words are used as features. For example, in Figure 2, consider target first-word features extracted from an inverted reordering example (Xiong et al., 2006) in MEBTG, in which first words on two blocks are both "the". This kind of feature set is too common and not representative enough to predict the correct orientation. Intuitively, one solution is to extend the feature set by considering both boundary words, forming a more complete boundary description. However, this method is still based on lexicalized features, which causes data sparseness problem and fails to generalize. In Figure 2, for example, the orientation should basically be the same, when the source/target words "計畫/plan" from block A_1 is replaced by other similar nouns and translations (e.g. "plans", "events" or "meetings"). However, such features would be treated as unseen by the current ME model, since the training data can not possibly cover all such similar cases.

In this paper we present an improved reordering model based on BTG, with bilingual linguistic features from neighboring blocks. To avoid data sparseness problem, both source and target words are classified; we perform part-of-speech (POS) tagging on source language, and word classification on target one, as shown in Figure 2. Additionally, features are extracted and classified depending on lengths of blocks in order to obtain a more informed model.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 describes the model used in our BTG-based SMT systems. Section 4 formally describes our bilingual linguistic reordering model. Section 5 and Section 6 explain the implementation of our systems. We show the experimental results in Section 7 and make the conclusion in Section 8.

2 Related Work

In statistical machine translation, reordering model is concerned with predicting correct orders of target language sentence given a source language one and translation pairs. For example, in phrase-based SMT systems (Koehn et al., 2003; Koehn, 2004), distortion model is used, in which reordering probabilities depend on relative positions of target side phrases between adjacent blocks. However, distortion model can not model long-distance reordering, due to the lack of context information, thus is difficult to predict correct orders under different circumstances. Therefore, while phrase-based SMT moves from words to phrases as the basic unit of translation, implying effective local reordering within phrases, it suffers when determining phrase reordering, especially when phrases are longer than three words (Koehn et al., 2003).

There have been much effort made to improve reordering model in SMT. For example, researchers have been studying CKY parsing over the last decade, which considers translations and orientations of two neighboring block according to grammar rules or context information. In hierarchical phrase-based systems (Chiang, 2005), for example, SCFG rules are automatically learned from aligned bilingual corpus, and are applied in CKY style decoding.

As an another application of CKY parsing technique is BTG-based SMT. Xiong et al. (2006) and Xiong et al. (2008a) developed MEBTG systems, in which first or tail words from reordering examples are used as features to train ME-based reordering models.

Similarly, Zhang et al. (2007) proposed a model similar to BTG, which uses first/tail words of phrases, and syntactic labels (e.g. NP and VP)

from source parse trees as features. In their work, however, inverted rules are allowed to apply only when source phrases are syntactic; for non-syntactic ones, blocks are combined straight with a constant score.

More recently, Xiong et al. (2008b) proposed LABTG, which incorporates linguistic knowledge by adding features such as syntactic labels and POS from source trees to improve their MEBTG. Different from Zhang's work, their model do not restrict non-syntactic phrases, and applies inverted rules on any pair of neighboring blocks.

Although POS information is used in LABTG and Zhang's work, their models are syntax-oriented, since they focus on syntactic labels. Boundary POS is considered in LABTG only when source phrases are not syntactic phrases.

In contrast to the previous works, we present a reordering model for BTG that uses bilingual information including class-level features of POS and word classes. Moreover, our model is dedicated to boundary features and considers different combinations of phrase lengths, rather than only first/tail words. In addition, current state-of-the-art Chinese parsers, including the one used in LABTG (Xiong et al., 2005), lag beyond in inaccuracy, compared with English parsers (Klein and Manning, 2003; Petrov and Klein 2007). In our work, we only use more reliable information such as Chinese word segmentation and POS tagging (Ma and Chen, 2003).

3 The Model

Following Wu (1996) and Xiong et al. (2006), we implement BTG-based SMT as our system, in which three rules are applied during decoding:

$$A \rightarrow [A_1 \quad A_2] \tag{1}$$

$$A \rightarrow \langle A_1 \quad A_2 \rangle \tag{2}$$

$$A \rightarrow x / y \tag{3}$$

where A_1 and A_2 are blocks in source order. Straight rule (1) and inverted rule (2) are reordering rules. They are applied for predicting target-side order when combining two blocks, and form the reordering model with the distributions

$$\mathrm{P}_{\mathrm{reo}}(A_1, A_2, order)^{\lambda_{reo}}$$

where $order \in \{straight, inverted\}$.

In MEBTG, a ME reordering model is trained using features extracted from reordering examples of aligned parallel corpus. First words on neighboring blocks are used as features. In reordering example (a), for example, the feature set is

{"S1L=three", "S2L=ago", "T1L=3", "T2L=前"}

where "S1" and "T1" denote source and target phrases from the block A_1.

Rule (3) is lexical translation rule, which translates source phrase x into target phrase y. We use the same feature functions as typical phrase-based SMT systems (Koehn et al., 2005):

$$\mathrm{P}_{\mathrm{trans}}(x \mid y) = p(x \mid y)^{\lambda_1} \cdot p(y \mid x)^{\lambda_2} \cdot p_{lw}(x \mid y)^{\lambda_3} \cdot p_{lw}(y \mid x)^{\lambda_4} \cdot \mathrm{e}^{\lambda_5} \cdot \mathrm{e}^{|y|\lambda_6}$$

where $p_{lw}(x \mid y)^{\lambda_3} \cdot p_{lw}(y \mid x)^{\lambda_4}$, e^{λ_5} and $\mathrm{e}^{|y|\lambda_6}$ are lexical translation probabilities in both directions, phrase penalty and word penalty.

During decoding, the blocks are produced by applying either one of two reordering rules on two smaller blocks, or applying lexical rule (3) on some source phrase. Therefore, the score of a block A is defined as

$$\mathrm{P}(A) = \mathrm{P}(A_1) \cdot \mathrm{P}(A_2) \cdot \Delta\mathrm{P}_{\mathrm{lm}}(A_1, A_2)^{\lambda_{lm}} \cdot \mathrm{P}_{\mathrm{reo}}(A_1, A_2, order)^{\lambda_{reo}}$$

or

$$\mathrm{P}(A) = \mathrm{P}_{\mathrm{lm}}(A)^{\lambda_{lm}} \cdot \mathrm{P}_{\mathrm{trans}}(x \mid y)$$

where $\mathrm{P}_{\mathrm{lm}}(A)^{\lambda_{lm}}$ and $\Delta\mathrm{P}_{\mathrm{lm}}(A_1, A_2)^{\lambda_{lm}}$ are respectively the usual and incremental score of language model.

To tune all lambda weights above, we perform minimum error rate training (Och, 2003) on the development set described in Section 7.

Let B be the set of all blocks with source side sentence C. Then the best translation of C is the target side of the block $\overline{A}$, where

$$\overline{A} = \underset{A \in B}{\operatorname{argmax}}\ P(A)$$

4 Bilingual Linguistic Model

In this section, we formally describe the problem we want to address and the proposed method.

4.1 Problem Statement

We focus on extracting features representative of the two neighboring blocks being considered for reordering by the decoder, as described in Section 3. We define $S(A)$ and $T(A)$ as the information on source and target side of a block A. For two neighboring blocks A_1 and A_2, the set of features extracted from information of them is denoted as feature set function $F(S(A_1), S(A_2), T(A_1), S(A_2))$. In Figure 1 (b), for example, $S(A_1)$ and $T(A_1)$ are simply the both sides sentences "3 年" and "three years", and $F(S(A_1), S(A_2), T(A_1), S(A_2))$ is

{"S1L=three", "S2L=after", "T1L=3", "T2L=後"}

where "S1L" denotes the first source word on the block A_1, and "T2L" denotes the first target word on the block A_2.

Given the adjacent blocks A_1 and A_2, our goal includes (1) adding more linguistic and representative information to A_1 and A_2 and (2) finding a feature set function F' based on added linguistic information in order to train a more linguistically motivated and effective model.

4.2 Word Classification

As described in Section 1, designing a more complete feature set causes data sparseness problem, if we use lexical features. One natural solution is using POS and word class features.

In our model, we perform Chinese POS tagging on source language. In Xiong et al. (2008b) and Zhang et al. (2007), Chinese parsers with Penn Chinese Treebank (Xue et al., 2005) style are used to derive source parse trees, from which source-side features such as POS are extracted. However, due to the relatively low accuracy of current Chinese parsers compared with English ones, we instead use CKIP Chinese word segmentation system (Ma and Chen, 2003) in order to derive Chinese tags with high accuracy. Moreover, compared with the Treebank Chinese tagset, the CKIP tagset provides more fine-grained tags, including many tags with semantic information (e.g., Nc for place nouns, Nd for time nouns), and verb transitivity and subcategorization (e.g., VA for intransitive verbs, VC for transitive verbs, VK for verbs that take a clause as object).

On the other hand, using the POS features in combination with the lexical features in target language will cause another sparseness problem in the phrase table, since one source phrase would map to multiple target ones with different POS sequences.

As an alternative, we use mkcls toolkit (Och, 1999), which uses maximum-likelihood principle to perform classification on target side. After classification, the toolkit produces a many-to-one mapping between English tokens and class numbers. Therefore, there is no ambiguity of word class in target phrases and word class features can be used independently to avoid data sparseness problem and the phrase table remains unchanged.

As mentioned in Section 1, features based on words are not representative enough in some cases, and tend to cause sparseness problem. By classifying words we are able to linguistically generalize the features, and hence predict the rules more robustly. In Figure 2, for example, the target words are converted to corresponding classes, and form the more complete boundary feature set

{"T1L=14", "T1R=18", "T2L=14", "T2R=50"} (4)

In the feature set (4), #14 is the class containing "the", #18 is the class containing "plans", and #50 is the class containing "of." Note that we add last-word features "T1R=18" and "T2R=50". As mentioned in Section 1, the word "plan" from block A_1 is replaceable with similar nouns. This extends to other nominal word classes to realize the general rule of inverting "the ... NOUN" and "the ... of".

It is hard to achieve this kind of generality using only lexicalized feature. With word classification, we gather feature sets with similar concepts from the training data. Table 1 shows the word classes can be used effectively to cope with data sparseness. For example, the feature set (4) occurs 309 times in our training data, and only 2 of them are straight, with the remaining 307 inverted examples, implying that similar features based on word classes lead to similar orientation. Additional examples of similar feature sets with different word classes are shown in Table 1.

class X	T1R = X	straight/inverted
9	graph, government	2/488
18	plans, events	2/307
20	bikes, motors	0/694
48	day, month, year	4/510

Table 1: List of feature sets in the form of {"T1L=14", "T1R=*X*", "T2L=14", "T2R=50"}.

4.3 Feature with Length Consideration

Boundary features using both the first and last words provide more detailed descriptions of neighboring blocks. However, we should take the special case blocks with length 1 into consideration. For example, consider two features sets from straight and inverted reordering examples (a) and (b) in Figure 3. There are two identical source features in both feature set, since first words on block A_1 and last words on block A_2 are the same:

$$\{\text{"S1L=P","S2R=Na"}\} \subseteq F(S(A_1),S(A_2),T(A_1),\ S(A_2))$$

Therefore, without distinguishing the special case, the features would represent quite different cases with the same feature, possibly leading to failure to predict orientations of two blocks.

We propose a method to alleviate the problem of features with considerations of lengths of two adjacent phrases by classifying both the both source and target phrase pairs into one of four classes: M, L, R and B, corresponding to different combinations of phrase lengths.

Suppose we are given two neighboring blocks A_1 and A_2, with source phrases P_1 and P_2 respectively. Then the feature set from source side is classified into one of the classes as follows. We give examples of feature set for each class according to Figure 4.

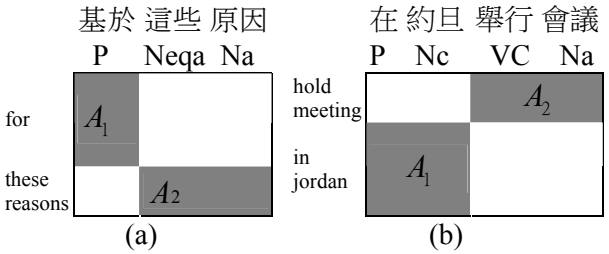


Figure 3: Two reordering examples with ambiguous features on source side.

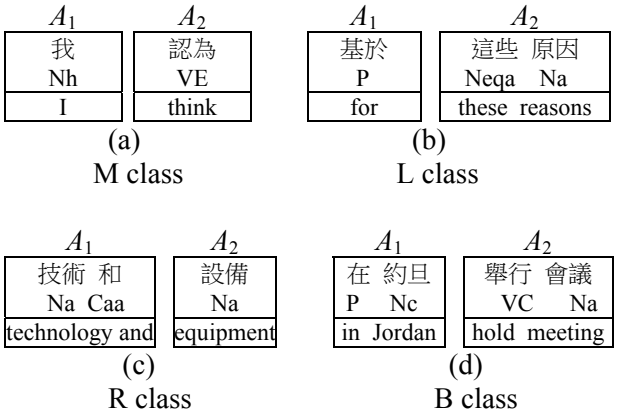


Figure 4: Examples of different length combinations, mapping to four classes.

1. M class. The lengths of P_1 and P_2 are both 1. In Figure 4 (a), for example, the feature set is

   {"M1=Nh", "M2=VE"}

2. L class. The length of P_1 is 1, and the length of P_2 is greater than 1. In Figure 4 (b), for example, the feature set is

   {"L1=P", "L2=Neqa", "L3=Na"}

3. R class. The length of P_1 is greater than 1, and the length of P_2 is 1. In Figure 4 (c), for example, the feature set is

   {"R1=Na", "R2=Caa", "R3=Na"}

4. B class. The lengths of P_1 and P_2 are both greater than 1. In Figure 4 (d), for example, the feature set is

   {"B1=P", "B2=Nc", "B3=VC", "B4=Na"}

We use the same scheme to classify the two target phrases. Since both source and target words are classified as described in Section 4.2, the feature sets are more representative and tend to lead to consistent prediction of orientation. Additionally, the length-based features are easy to fit into memory, in contrast to lexical features in MEBTG.

To summarize, we extract features based on word lengths, target-language word classes, and fine-grained, semantic oriented parts of speech. To illustrate, we use the neighboring blocks from Fig-

ure 2 to show an example of complete bilingual linguistic feature set:

{"S.B1=Nes", "S.B2=Nv", "S.B3=DE",
"S.B4=Na", "T.B1=14", "T.B2=18", "T.B3=14",
"T.B4=50"}

where "S." and "T." denote source and target sides.

In the next section, we describe the process of preparing the feature data and training an ME model. In Section 7, we perform evaluations of this ME-based reordering model against standard phrase-based SMT and previous work based on ME and BTG.

5 Training

In order to train the translation and reordering model, we first set up Moses SMT system (Koehn et al., 2007). We obtain aligned parallel sentences and the phrase table after the training of Moses, which includes running GIZA++ (Och and Ney, 2003), grow-diagonal-final symmetrization and phrase extraction (Koehn et al., 2005). Our system shares the same translation model with Moses, since we directly use the phrase table to apply translation rules (3).

On the other side, we use the aligned parallel sentences to train our reordering model, which includes classifying words, extracting bilingual phrase samples with orientation information, and training an ME model for predicting orientation.

To perform word classification, the source sentences are tagged and segmented before the Moses training. As for target side, we ran the Moses scripts to classify target language words using the mkcls toolkit before running GIZA++. Therefore, we directly use its classification result, which generate 50 classes with 2 optimization runs on the target sentences.

To extract the reordering examples, we choose sentence pairs with top 50% alignment scores provided by GIZA++, in order to fit into memory. Then the extraction is performed on these aligned sentence pairs, together with POS tags and word classes, using basically the algorithm presented in Xiong et al. (2006). However, we enumerate all reordering examples, rather than only extract the smallest straight and largest inverted examples. Finally, we use the toolkit by Zhang (2004) to train the ME model with extracted reordering examples.

6 Decoding

We develop a bottom-up CKY style decoder in our system, similar to Chiang (2005). For a Chinese sentence *C*, the decoder finds its best translation on the block with entire *C* on source side. The decoder first applies translation rules (3) on cells in a CKY matrix. Each cell denotes a sequence of source phrases, and contains all of the blocks with possible translations. The longest length of source phrase to be applied translations rules is restricted to 7 words, in accordance with the default settings of Moses training scripts.

To reduce the search space, we apply threshold pruning and histogram pruning, in which the block scoring worse than 10^{-2} times the best block in the same cell or scoring worse than top 40 highest scores would be pruned. These pruning techniques are common in SMT systems. We also apply recombination, which distinguish blocks in a cell only by 3 leftmost and rightmost target words, as suggested in (Xiong et al., 2006).

7 Experiments and Results

We perform Chinese-to-English translation task on NIST MT-06 test set, and use Moses and MEBTG as our competitors.

The bilingual training data containing 2.2M sentences pairs from Hong Kong Parallel Text (LDC2004T08) and Xinhua News Agency (LDC2007T09), with length shorter than 60, is used to train the translation and reordering model. The source sentences are tagged and segmented with CKIP Chinese word segmentation system (Ma and Chen, 2003).

About 35M reordering examples are extracted from top 1.1M sentence pairs with higher alignment scores. We generate 171K features for lexicalized model used in MEBTG system, and 1.41K features for our proposed reordering model.

For our language model, we use Xinhua news from English Gigaword Third Edition (LDC2007T07) to build a trigram model with SRILM toolkit (Stolcke, 2002).

Our development set for running minimum error rate training is NIST MT-08 test set, with sentence lengths no more than 20. We report the experimental results on NIST MT-06 test set. Our evaluation metric is BLEU (Papineni et al., 2002) with case-insensitive matching from unigram to four-gram.

System	BLEU-4
Moses(distortion)	22.55
Moses(lexicalized)	23.42
MEBTG	23.65
WC+LC	24.96

Table 2: Performances of various systems.

The overall result of our experiment is shown in Table 2. The lexicalized MEBTG system proposed by Xiong et al. (2006) uses first words on adjacent blocks as lexical features, and outperforms phrase-based Moses with default distortion model and enhanced lexicalized model, by 1.1 and 0.23 BLEU points respectively. This suggests lexicalized Moses and MEBTG with context information outperforms distance-based distortion model. Besides, MEBTG with structure constraints has better global reordering estimation than unstructured Moses, while incorporating their local reordering ability by using phrase tables.

The proposed reordering model trained with word classification (WC) and length consideration (LC) described in Section 4 outperforms MEBTG by 1.31 point. This suggests our proposed model not only reduces the model size by using 1% fewer features than MEBTG, but also improves the translation quality.

We also evaluate the impacts of WC and LC separately and show the results in Table 3-5. Table 3 shows the result of MEBTG with word classified features. While classified MEBTG only improves 0.14 points over original lexicalized one, it drastically reduces the feature size. This implies WC alleviates data sparseness by generalizing the observed features.

Table 4 compares different length considerations, including boundary model demonstrated in Section 4.2, and the proposed LC in Section 4.3. Although boundary model describes features better than using only first words, which we will show later, it suffers from data sparseness with twice feature size of MEBTG. The LC model has the largest feature size but performs best among three systems, suggesting the effectiveness of our LC.

In Table 5 we show the impacts of WC and LC together. Note that all the systems with WC significantly reduce the size of features compared to lexicalized ones.

System	Feature size	BLEU-4
MEBTG	171K	23.65
WC+MEBTG	0.24K	23.79

Table 3: Performances of lexicalized and word classified MEBTG.

System	Feature size	BLEU-4
MEBTG	171K	23.65
Boundary	349K	23.42
LC	780K	23.86

Table 4: Performances of BTG systems with different representativeness.

System	Feature size	BLEU-4
MEBTG	171K	23.65
WC+MEBTG	0.24K	23.79
WC+Bounary	0.48K	24.29
WC+LC	1.41K	24.96

Table 5: Different representativeness with word classification.

While boundary model is worse than first-word MEBTG in Table 4, it outperforms the latter when both are performed WC. We obtain the best result that outperforms the baseline MEBTG by more than 1 point when we apply WC and LC together.

Our experimental results show that we are able to ameliorate the sparseness problem by classifying words, and produce more representative features by considering phrase length. Moreover, they are both important, in that we are unable to outperform our competitors by a large margin unless we combine both WC and LC. In conclusion, while designing more representative features of reordering model in SMT, we have to find solutions to generalize them.

8 Conclusion and Future Works

We have proposed a bilingual linguistic reordering model to improve current BTG-based SMT systems, based on two drawbacks of previously proposed reordering model, which are sparseness and representative problem.

First, to solve the sparseness problem in previously proposed lexicalized model, we perform word classification on both sides.

Secondly, we present a more representative feature extraction method. This involves considering length combinations of adjacent phrases.

The experimental results of Chinese-to-English task show that our model outperforms baseline phrase-based and BTG systems.

We will investigate more linguistic ways to classify words in future work, especially on target language. For example, using word hierarchical structures in WordNet (Fellbaum, 1998) system provides more linguistic and semantic information than statistically-motivated classification tools.

Acknowledgements

This work was supported by National Science Council of Taiwan grant NSC 95-2221-E-007-182-MY3.

References

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

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

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT/NAACL 2003*.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrased-Based Statistical Machine Translation Models. In *Proceedings of AMTA 2004*.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *International Workshop on Spoken Language Translation*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan,Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constrantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL 2007, Demonstration Session*.

Dan Klein and Christopher D. Manning. 2003. *Accurate Unlexicalized Parsing*. In *Proceedings of ACL 2003*.

Shankar Kumar and William Byrne. 2005. *Local phrase reordering models for statistical machine translation*. In *Proceedings of HLT-EMNLP 2005*.

Wei-Yun Ma and Keh-Jiann Chen. 2003. Introduction to CKIP Chinese Word Segmentation System for the First International Chinese Word Segmentation Bakeoff. In *Proceedings of ACL, Second SIGHAN Workshop on Chinese Language Processing*, pp168-171.

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *EACL '99: Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 71–76, Bergen, Norway, June.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29:19-51.

Franz Josef 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 the 40th Annual Meeting of the ACL*, pages 311–318.

Slav Petrov and Dan Klein. 2007. Improved Inference-for Unlexicalized Parsing. In *Proceedings of HLT-NAACL 2007*.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Dekai Wu. 1996. A Polynomial-Time Algorithm for Statistical Machine Translation. In *Proceedings of ACL 1996*.

Deyi Xiong, Shuanglong Li, Qun Liu, Shouxun Lin, and Yueliang Qian. 2005. Parsing the Penn Chinese treebank with semantic knowledge. In *Proceedings of IJCNLP 2005*, pages 70-81.

Deyi Xiong, Qun Liu and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *Proceedings of ACL-COLING 2006*.

Deyi Xiong, Min Zhang, Aiti Aw, Haitao Mi, Qun Liu, and Shouxun Liu. 2008a. Refinements in BTG-based statistical machine translation. In *Proceedings of IJCNLP 2008*, pp. 505-512.

Deyi Xiong, Min Zhang, Ai Ti Aw, and Haizhou Li. 2008b. Linguistically Annotated BTG for Statistical Machine Translation. In *Proceedings of COLING 2008*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase

structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

R. Zens, H. Ney, T. Watanabe, and E. Sumita. 2004. Reordering Constraints for Phrase-Based Statistical Machine Translation. In *Proceedings of CoLing 2004*, Geneva, Switzerland, pp. 205-211.

Le Zhang. 2004. Maximum Entropy Modeling Toolkit for Python and C++. Available at http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

Dongdong Zhang, Mu Li, Chi-Ho Li and Ming Zhou. 2007. Phrase Reordering Model Integrating Syntactic Knowledge for SMT. In *Proceedings of EMNLP-CoNLL 2007*.

May All Your Wishes Come True: A Study of Wishes and How to Recognize Them

Andrew B. Goldberg, Nathanael Fillmore, David Andrzejewski
Zhiting Xu, Bryan Gibson, Xiaojin Zhu
Computer Sciences Department, University of Wisconsin-Madison, Madison, WI 53706, USA
{goldberg, nathanae, andrzeje, zhiting, bgibson, jerryzhu}@cs.wisc.edu

Abstract

A wish is "a desire or hope for something to happen." In December 2007, people from around the world offered up their wishes to be printed on confetti and dropped from the sky during the famous New Year's Eve "ball drop" in New York City's Times Square. We present an in-depth analysis of this collection of wishes. We then leverage this unique resource to conduct the first study on building general "wish detectors" for natural language text. Wish detection complements traditional sentiment analysis and is valuable for collecting business intelligence and insights into the world's wants and desires. We demonstrate the wish detectors' effectiveness on domains as diverse as consumer product reviews and online political discussions.

1 Introduction

Each year, New York City rings in the New Year with the famous "ball drop" in Times Square. In December 2007, the Times Square Alliance, co-producer of the Times Square New Year's Eve Celebration, launched a Web site called the Virtual Wishing Wall[1] that allowed people around the world to submit their New Year's wishes. These wishes were then printed on confetti and dropped from the sky at midnight on December 31, 2007 in sync with the ball drop.

We obtained access to this set of nearly 100,000 New Year's wishes, which we call the "WISH corpus." Table 1 shows a selected sample of the WISH corpus. Some are far-reaching fantasies and aspirations, while others deal with everyday concerns like economic and medical distress. We analyze this first-of-its-kind corpus in Section 2.

The New Oxford American Dictionary defines "wish" as "a desire or hope for something to happen." How wishes are expressed, and how such wishful expressions can be automatically recognized, are open questions in natural language processing. Leveraging the WISH corpus, we conduct the first study on building general "wish detectors" for natural language text, and demonstrate their effectiveness on domains as diverse as consumer product reviews and online political discussions. Such wish detectors have tremendous value in collecting business intelligence and public opinions. We discuss the wish detectors in Section 3, and experimental results in Section 4.

1.1 Relation to Prior Work

Studying wishes is valuable in at least two aspects:

1. Being a special genre of subjective expression, wishes add a novel dimension to sentiment analysis. Sentiment analysis is often used as an automatic market research tool to collect valuable business intelligence from online text (Pang and Lee, 2008; Shanahan et al., 2005; Koppel and Shtrimberg, 2004; Mullen and Malouf, 2008). Wishes differ from the recent focus of sentiment analysis, namely opinion mining, by revealing what people explicitly want to happen, not just what they like or dislike (Ding et al., 2008; Hu and Liu, 2004). For example, wishes in product reviews could contain new feature requests. Consider the following (real) prod-

[1] http://www.timessquarenyc.org/nye/nye_interactive.html

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 263–271,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

514	*peace on earth*
351	*peace*
331	*world peace*
244	*happy new year*
112	*love*
76	*health and happiness*
75	*to be happy*
51	*i wish for world peace*
21	*i wish for health and happiness for my family*
21	*let there be peace on earth*
16	*i wish u to call me if you read this 555-1234*
16	*to find my true love*
8	*i wish for a puppy*
7	*for the war in iraq to end*
6	*peace on earth please*
5	*a free democratic venezuela*
5	*may the best of 2007 be the worst of 2008*
5	*to be financially stable*
1	*a little goodness for everyone would be nice*
1	*i hope i get accepted into a college that i like*
1	*i wish to get more sex in 2008*
1	*please let* $name$ *be healthy and live all year*
1	*to be emotionally stable and happy*
1	*to take over the world*

Table 1: Example wishes and their frequencies in the WISH corpus.

uct review excerpt: "*Great camera. Indoor shots with a flash are not quite as good as 35mm. I wish the camera had a higher optical zoom so that I could take even better wildlife photos.*" The first sentence contains positive opinion, the second negative opinion. However, wishful statements like the third sentence are often annotated as non-opinion-bearing in sentiment analysis corpora (Hu and Liu, 2004; Ding et al., 2008), even though they clearly contain important information. An automatic "wish detector" text-processing tool can be useful for product manufacturers, advertisers, politicians, and others looking to discover what people want.

2. Wishes can tell us a lot about people: their innermost feelings, perceptions of what they're lacking, and what they desire (Speer, 1939). Many psychology researchers have attempted to quantify the contents of wishes and how they vary with factors such as location, gender, age, and personality type (Speer, 1939; Milgram and Riedel, 1969; Ehrlichman and Eichenstein, 1992; King and Broyles, 1997). These studies have been small scale with only dozens or hundreds of participants. The WISH corpus provides the first large-scale collection of wishes as a window into the world's desires.

Beyond sentiment analysis, classifying sentences as wishes is an instance of non-topical classification. Tasks under this heading include computational humor (Mihalcea and Strapparava, 2005), genre classification (Boese and Howe, 2005), authorship attribution (Argamon and Shimoni, 2003), and metaphor detection (Krishnakumaran and Zhu, 2007), among others (Mishne et al., 2007; Mihalcea and Liu, 2006). We share the common goal of classifying text into a unique set of target categories (in our case, wishful and non-wishful), but use different techniques catered to our specific task. Our feature-generation technique for wish detection resembles template-based methods for information extraction (Brin, 1999; Agichtein and Gravano, 2000).

2 Analyzing the WISH Corpus

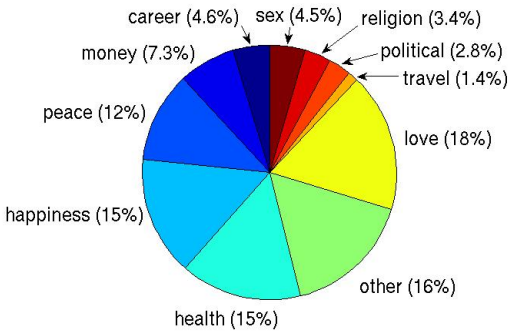
We analyze the WISH corpus with a variety of statistical methods. Our analyses not only reveal what people wished for on New Year's Eve, but also provide insight for the development of wish detectors in Section 3.

The complete WISH corpus contains nearly 100,000 wishes collected over a period of 10 days in December 2007, most written in English, with the remainder in Portuguese, Spanish, Chinese, French, and other languages. For this paper, we consider only the 89,574 English wishes. Most of these English wishes contain optional geographic meta data provided by the wisher, indicating a variety of countries (not limited to English-speaking) around the world. We perform minimal preprocessing, including TreeBank-style tokenization, downcasing, and punctuation removal. Each wish is treated as a single entity, regardless of whether it contains multiple sentences. After preprocessing, the average length of a wish is 8 tokens.

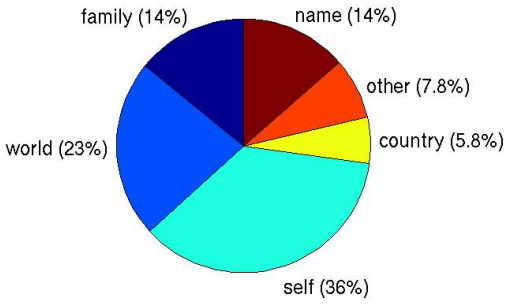
2.1 The Topic and Scope of Wishes

As a first step in understanding the content of the wishes, we asked five annotators to manually annotate a random subsample of 5,000 wishes. Sections 2.1 and 2.2 report results on this subsample.

The wishes were annotated in terms of two at-



(a) Topic of Wishes



(b) Scope of Wishes

Figure 1: Topic and scope distributions based on manual annotations of a random sample of 5,000 wishes in the WISH corpus.

tributes: topic and scope. We used 11 pre-defined topic categories, and their distribution in this subsample of the WISH corpus is shown in Figure 1(a). The most frequent topic is *love*, while *health*, *happiness*, and *peace* are also common themes. Many wishes also fell into an *other* category, including specific individual requests ("i wish for a new puppy"), solicitations or advertisements ("call me 555-1234", "visit $website$.com"), or sinister thoughts ("to take over the world").

The 5,000 wishes were also manually assigned a scope. The scope of a wish refers to the range of people that are targeted by the wish. We used 6 pre-defined scope categories: *self* ("I want to be happy"), *family* ("For a cure for my husband"), specific person by *name* ("Prayers for $name$"), *country* ("Bring our troops home!"), *world* ("Peace to everyone in the world"), and *other*. In cases where multiple scope labels applied, the broadest scope was selected. Figure 1(b) shows the scope distribution. It is bimodal: over one third of the wishes are narrowly directed at one's self, while broad wishes at the world level are also frequent. The in-between scopes are less frequent.

2.2 Wishes Differ by Geographic Location

As mentioned earlier, wishers had the option to enter a city/country when submitting wishes. Of the manually annotated wishes, about 4,000 included valid location information, covering all 50 states in the U.S., and all continents except Antarctica.

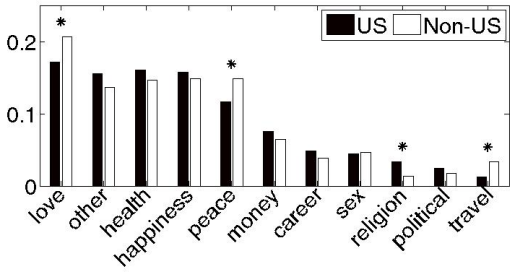
We noticed a statistically significant difference between wishes submitted from the United States (about 3600) versus non-U.S. (about 400), both in terms of their topic and scope distributions. For each comparison, we performed a Pearson χ^2-test using location as the explanatory variable and either topic or scope as the response variable.[2] The null hypothesis is that the variables are independent. For both tests we reject the null hypothesis, with $p < 0.001$ for topic, and $p = 0.006$ for scope. This indicates a dependence between location and topic/scope. Asterisks in Figure 2 denote the labels that differ significantly between U.S. and non-U.S. wishes.[3]

In particular, we observed that there are significantly more wishes about *love*, *peace*, and *travel* from non-U.S. locales, and more about *religion* from the U.S. There are significantly more *world*-scoped wishes from non-U.S. locales, and more *country*- and *family*-scoped wishes from the U.S.

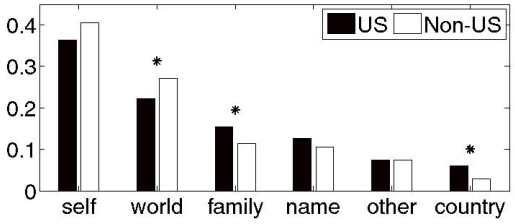
We also compared wishes from "red states" versus "blue states" (U.S. states that voted a majority for the Republican and Democratic presidential candidates in 2008, respectively), but found no significant differences.

[2]The topic test examined a 2×11 contingency table, while the scope test used a 2×6 contingency table. In both tests, all of the cells in the tables had an expected frequency of at least 5, so the χ^2 approximation is valid.

[3]To identify the labels that differ significantly by location, we computed the standardized residuals for the cells in the two contingency tables. Standardized residuals are approximately $\mathcal{N}(0,1)$-distributed and can be used to locate the major contributors to a significant χ^2-test statistic (Agresti, 2002). The asterisks in Figure 2 indicate the surprisingly large residuals, i.e., the difference between observed and expected frequencies is outside a 95% confidence interval.



(a) Wish topics differ by Location



(b) Wish scopes differ by Location

Figure 2: Geographical breakdown of topic and scope distributions based on approximately 4,000 location-tagged wishes. Asterisks indicate statistically significant differences.

2.3 Wishes Follow Zipf's Law

We now move beyond the annotated subsample and examine the full set of 89,574 English wishes. We noticed that a small fraction (4%) of unique wishes account for a relatively large portion (16%) of wish occurrences, while there are also many wishes that only occur once. The question naturally arises: do wishes obey Zipf's Law (Zipf, 1932; Manning and Schütze, 1999)? If so, we should expect the frequency of a unique wish to be inversely proportional to its rank, when sorted by frequency. Figure 3 plots rank versus frequency on a log-log scale and reveals an approximately linear negative slope, thus suggesting that wishes do follow Zipf's law. It also shows that low-occurrence wishes dominate, hence learning might be hindered by data sparseness.

2.4 Latent Topic Modeling for Wishes

The 11 topics in Section 2.1 were manually predefined based on domain knowledge. In contrast, in this section we applied Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to identify the latent topics in the full set of 89,574 English wishes in an unsupervised fashion. The goal is to validate and complement the study in Section 2.1.

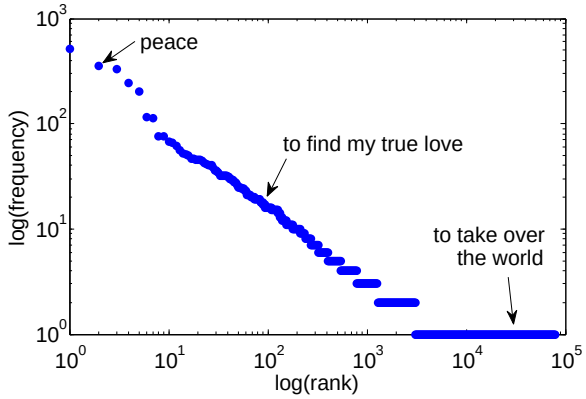


Figure 3: The rank vs. frequency plot of wishes, approximately obeying Zipf's law. Note the log-log scale.

To apply LDA to the wishes, we treated each individual wish as a short document. We used 12 topics, Collapsed Gibbs Sampling (Griffiths and Steyvers, 2004) for inference, hyperparameters $\alpha = 0.5$ and $\beta = 0.1$, and ran Markov Chain Monte Carlo for 2000 iterations.

The resulting 12 LDA topics are shown in Table 2, in the form of the highest probability words $p(\text{word}|\text{topic})$ in each topic. We manually added summary descriptors for readability. With LDA, it is also possible to observe which words were assigned to which topics in each wish. For example, LDA assigned most words in the wish "world(8) peace(8) and my friends(4) in iraq(1) to come(1) home(1)" to two topics: peace and troops (topic numbers in parentheses). Interestingly, these LDA topics largely agree with the pre-defined topics in Section 2.1.

3 Building Wish Detectors

We now study the novel NLP task of wish detection, i.e., classifying individual sentences as being wishes or not. Importantly, we want our approach to transfer to domains other than New Year's wishes, including consumer product reviews and online political discussions. It should be pointed out that wishes are highly domain dependent. For example, "I wish for world peace" is a common wish on New Year's Eve, but is exceedingly rare in product reviews; and vice versa: "I want to have instant access to the volume" may occur in product reviews, but is an un-

Topic	Summary	Top words in the topic, sorted by $p(\text{word}\|\text{topic})$
0	New Year	year, new, happy, 2008, best, everyone, great, years, wishing, prosperous, may, hope
1	Troops	all, god, home, come, may, safe, s, us, bless, troops, bring, iraq, return, 2008, true, dreams
2	Election	wish, end, no, more, 2008, war, stop, president, paul, not, ron, up, free, less, bush, vote
3	Life	more, better, life, one, live, time, make, people, than, everyone, day, wish, every, each
4	Prosperity	health, happiness, good, family, friends, all, love, prosperity, wealth, success, wish, peace
5	Love	love, me, find, wish, true, life, meet, want, man, marry, call, someone, boyfriend, fall, him
6	Career	get, wish, job, out, t, hope, school, better, house, well, want, back, don, college, married
7	Lottery	wish, win, 2008, money, want, make, become, lottery, more, great, lots, see, big, times
8	Peace	peace, world, all, love, earth, happiness, everyone, joy, may, 2008, prosperity, around
9	Religion	love, forever, jesus, know, loves, together, u, always, 2, 3, 4, much, best, mom, christ
10	Family	healthy, happy, wish, 2008, family, baby, life, children, long, safe, husband, stay, marriage
11	Health	com, wish, s, me, lose, please, let, cancer, weight, cure, mom, www, mother, visit, dad

Table 2: Wish topics learned from Latent Dirichlet Allocation. Words are sorted by $p(\text{word}|\text{topic})$.

likely New Year's wish. For this initial study, we do assume that there are some labeled training data in the target domains of interest.

To transfer the knowledge learned from the out-of-domain WISH corpus to other domains, our key insight is the following: while the content of wishes (e.g., "world peace") may not transfer across domains, the ways wishes are expressed (e.g., "I wish for ___") may. We call these expressions *wish templates*. Our novel contribution is an unsupervised method for discovering candidate templates from the WISH corpus which, when applied to other target domains, improve wish detection in those domains.

i wish ___
i hope ___
i want ___
hopefully ___
if only ___
would be better if ___
would like ___ if ___
___ should ___
would that ___
can't believe ___ didn't ___
don't believe ___ didn't ___
___ do want
i can has ___

Table 3: Manual templates for identifying wishes.

3.1 Two Simple Wish Detectors

Before describing our template discovery method, we first describe two simple wish detectors, which serve as baselines.

1. **[Manual]**: It may seem easy to locate wishes. Perhaps looking for sentences containing the phrases "i wish," "i hope," or some other simple patterns is sufficient for identifying the vast majority of wishes in a domain. To test this hypothesis, we asked two native English speakers (not the annotators, nor affiliated with the project; no exposure to any of the wish datasets) to come up with text patterns that might be used to express wishes. They were shown three dictionary definitions of "to wish (v)" and "wish (n)". They produced a ranked list of 13 templates; see Table 3. The underscore matches any string. These templates can be turned into a simple rule-based classifier: If part of a sentence matches one of the templates, the sentence is classified as a wish. By varying the depth of the list, one can produce different precision/recall behaviors. Overall, we expect [Manual] to have relatively high precision but low recall.

2. **[Words]**: Another simple method for detecting wishes is to train a standard word-based text classifier using the labeled training set in the target domain. Specifically, we represent each sentence as a binary word-indicator vector, normalized to sum to 1. We then train a linear Support Vector Machine (SVM). This method may have higher recall, but precision may suffer. For instance, the sentence "Her wish was carried out by her husband" is not a wish, but could be misclassified as one because of the word "wish."

Note that neither of the two baseline methods uses the WISH corpus.

3.2 Automatically Discovering Wish Templates

We now present our method to automatically discover high quality wish templates using the WISH corpus. The key idea is to exploit redundancy in how the same wish content is expressed. For example, as we see in Table 1, both "world peace" and "i wish for world peace" are common wishes. Similarly, both "health and happiness" and "i wish for health and happiness" appear in the WISH corpus. It is thus reasonable to speculate that "i wish for ___" is a good wish template. Less obvious templates can be discovered in this way, too, such as "let there be ___" from "peace on earth" and "let there be peace on earth."

We formalize this intuition as a bipartite graph, illustrated in Figure 4. Let $W = \{w_1, \ldots, w_n\}$ be the set of unique wishes in the WISH corpus. The bipartite graph has two types of nodes: content nodes C and template nodes T, and they are generated as follows. If a wish w_j (e.g., "i wish for world peace") contains another wish w_i (e.g., "world peace"), we create a content node $c_1 = w_i$ and a template node t_1 ="i wish for ___". We denote this relationship by $w_j = c_1 + t_1$. Note the order of c_1 and t_1 is insignificant, as how the two combine is determined by the underscore in t_1, and $w_j = t_1 + c_1$ is just fine. In addition, we place a directed edge from c_1 to t_1 with edge weight count(w_j), the frequency of wish w_j in the WISH corpus. Then, a template node appears to be a good one if many heavy edges point to it.

On the other hand, a template is less desirable if it is part of a content node. For example, when w_j ="health and happiness" and w_i ="health", we create the template t_2 ="___ and happiness" and the content node $c_3 = w_i$. If there is another wish w_k ="i wish for health and happiness", then there will be a content node $c_2 = w_j$. The template t_2 thus contains some content words (since it matches c_2), and may not generalize well in a new domain. We capture this by backward edges: if $\exists c' \in C$, and $\exists$ string s (s not necessarily in C or W) such that $c' = s + t$, we add a backward edge from t to c' with edge weight count(c').

Based on such considerations, we devised the following scheme for scoring templates:

$$\text{score}(t) = \text{in}(t) - \text{out}(t), \quad (1)$$

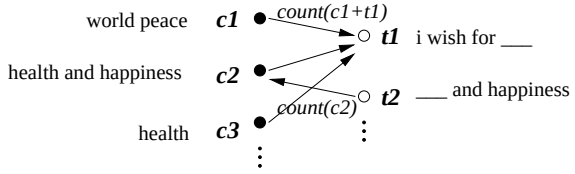


Figure 4: The bipartite graph to create templates.

where in(t) is the in-degree of node t, defined as the sum of edge weights coming into t; out(t) is the out-degree of node t, defined similarly. In other words, a template receives a high score if it is "used" by many frequent wishes but does not match many frequent content-only wishes. To create the final set of template features, we apply the threshold score$(t) \geq 5$. This produces a final list of 811 templates. Table 4 lists some of the top templates ranked by score(t). While some of these templates still contain time- or scope-related words ("for my family"), they are devoid of specific topical content. Notice that we have automatically identified several of the manually derived templates in Table 3, and introduce many new variations that a learning algorithm can leverage.

Top 10	Others in Top 200
___ in 2008	i want to ___
i wish for ___	___ for everyone
i wish ___	i hope ___
i want ___	my wish is ___
___ this year	please ___
i wish ___ in 2008	wishing for ___
i wish to ___	may you ___
___ for my family	i wish i had ___
i wish ___ this year	to finally ___
___ in the new year	for my family to have ___

Table 4: Top templates according to Equation 1.

3.3 Learning with Wish Template Features

After discovering wish templates as described above, we use them as features for learning in a new domain (e.g., product reviews). For each sentence in the new domain, we assign binary features indicating which templates match the sentence. Two types of matching are possible. *Strict matching* requires that the template must match an entire sentence from beginning to end, with at least one word filling in for the underscore. (All matching during the template generation process was strict.) *Non-strict matching*

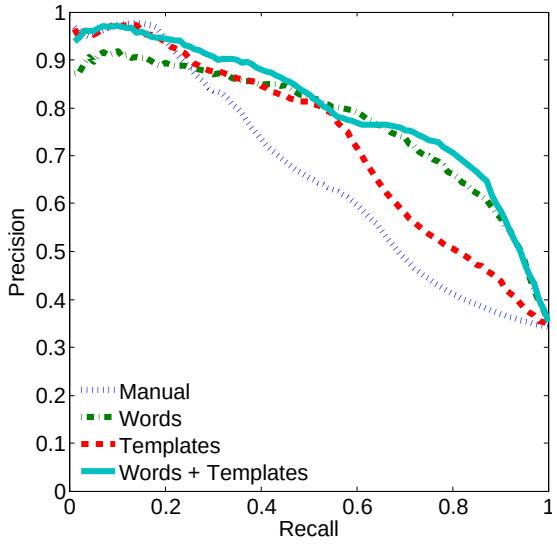


Figure 5: Politics domain precision-recall curves.

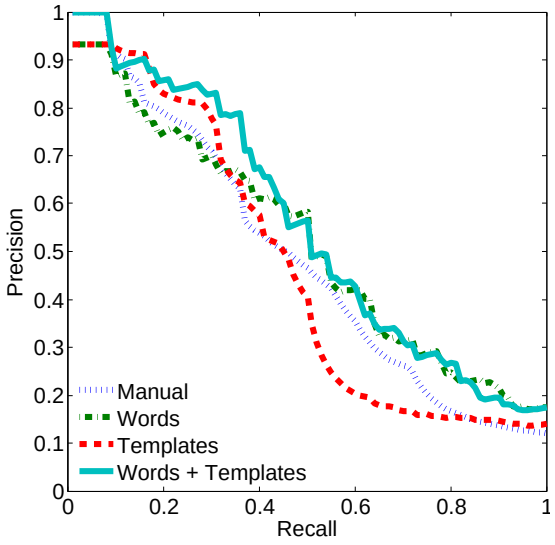


Figure 6: Products domain precision-recall curves.

requires only that template match somewhere within a sentence. Rather than choose one type of matching, we create both strict and non-strict template features (1622 binary features total) and let the machine learning algorithm decide what is most useful.

Our third wish detector, **[Templates]**, is a linear SVM with the 1622 binary wish template features. Our fourth wish detector, **[Words + Templates]**, is a linear SVM with both template and word features.

4 Experimental Results

4.1 Target Domains and Experimental Setup

We experimented with two domains, manually labeled at the sentence-level as wishes or non-wishes.[4] Example wishes are listed in Table 6.

Products. Consumer product reviews: 1,235 sentences selected from a collection of amazon.com and cnet.com reviews (Hu and Liu, 2004; Ding et al., 2008). 12% of the sentences are labeled as wishes.

Politics. Political discussion board postings: 6,379 sentences selected from politics.com (Mullen and Malouf, 2008). 34% are labeled as wishes.

We automatically split the corpora into sentences using MxTerminator (Reynar and Ratnaparkhi, 1997). As preprocessing before learning, we tokenized the text in the Penn TreeBank style, downcased, and removed all punctuation.

For all four wish detectors, we performed 10-fold cross validation. We used the default parameter in SVM^{light} for all trials (Joachims, 1999). As the data sets are skewed, we compare the detectors using precision-recall curves and the area under the curve (AUC). For the manual baseline, we produce the curve by varying the number of templates applied (in rank order), which gradually predicts more sentences as wishes (increasing recall at the expense of precision). A final point is added at recall 1.0, corresponding to applying an empty template that matches all sentences. For the SVM-based methods, we vary the threshold applied to the real-valued margin prediction to produce the curves. All curves are interpolated, and AUC measures are computed, using the techniques of (Davis and Goadrich, 2006).

4.2 Results

Figure 5 shows the precision-recall curves for the Politics corpus. All curves are averages over 10 folds (i.e., for each of 100 evenly spaced, interpolated recall points, the 10 precision values are averaged). As expected, [Manual] can be very precise with low recall—only the very top few templates achieve high precision and pick out a small number of wishes with "i wish" and "i hope." As we introduce more templates to cover more true wishes, precision drops off quickly. [Templates] is similar,

[4]These wish-annotated corpora are available for download at http://pages.cs.wisc.edu/~goldberg/wish_data.

Corpus	[Manual]	[Words]	[Templates]	[Words + Templates]
Politics	0.67 ± 0.03	0.77 ± 0.03	0.73 ± 0.03	0.80 ± 0.03
Products	0.49 ± 0.13	0.52 ± 0.16	0.47 ± 0.16	0.56 ± 0.16

Table 5: AUC results (10-fold averages $\pm$ one standard deviation).

Products:
the only area i wish apple had improved upon would be the screen
i just want music to eminate from it when i want how i want
the dial on the original zen was perfect and i wish it was on this model
i would like album order for my live albums and was just wondering
Politics:
all children should be allowed healthcare
please call on your representatives in dc and ask them to please stop the waste in iraq
i hope that this is a new beginning for the middle east
may god bless and protect the brave men and that we will face these dangers in the future

Table 6: Example target-domain wishes correctly identified by [Words + Templates].

with slightly better precision in low recall regions. [Words] is the opposite: bad in high recall but good in low recall regions. [Words + Templates] is the best, taking the best from both kinds of features to dominate other curves. Table 5 shows the average AUC across 10 folds. [Words + Templates] is significantly better than all other detectors under paired t-tests ($p = 1 \times 10^{-7}$ vs. [Manual], $p = 0.01$ vs. [Words], and $p = 4 \times 10^{-7}$ vs. [Templates]). All other differences are statistically significant, too.

Figure 6 shows the precision-recall curves for the Products corpus. Again, [Words + Templates] mostly dominates other detectors. In terms of average AUC across folds (Table 5), [Words + Templates] is also the best. However, due to the small size of this corpus, the AUC values have high variance, and the difference between [Words + Templates] and [Words] is not statistically significant under a paired t-test ($p = 0.16$).

Finally, to understand what is being learned in more detail, we take a closer look at the SVM models' weights for one fold of the Products corpus (Table 7). The most positive and negative features make intuitive sense. Note that [Words + Templates] seems to rely on templates for selecting wishes and words for excluding non-wishes. This partially explains the synergy of combining the feature types.

Sign	[Words]	[Templates]	[Words + Templates]
+	wish	i hope ___	hoping ___
+	hope	i wish ___	i hope ___
+	hopefully	hoping ___	i just want ___
+	hoping	i just want ___	i wish ___
+	want	i would like ___	i would like ___
-	money	family ___	micro
-	find	___ forever	about
-	digital	let me ___	fix
-	again	___ d	digital
-	you	___ for my dad	you

Table 7: Features with the largest magnitude weights in the SVM models for one fold of the Products corpus.

5 Conclusions and Future Work

We have presented a novel study of wishes from an NLP perspective. Using the first-of-its-kind WISH corpus, we generated domain-independent wish templates that improve wish detection performance across product reviews and political discussion posts. Much work remains in this new research area, including the creation of more types of features. Also, due to the difficulty in obtaining wish-annotated training data, we plan to explore semi-supervised learning for wish detection.

Acknowledgements We thank the Times Square Alliance for providing the WISH corpus, and the Wisconsin Alumni Research Foundation. AG is supported in part by a Yahoo! Key Technical Challenges Grant.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *In Proceedings of the 5th ACM International Conference on Digital Libraries*, pages 85–94.

Alan Agresti. 2002. *Categorical Data Analysis*. Wiley-Interscience, second edition.

Shlomo Argamon and Anat Rachel Shimoni. 2003. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17:401–412.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Elizabeth Sugar Boese and Adele Howe. 2005. Genre classification of web documents. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05), Poster paper*.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In *WebDB '98: Selected papers from the International Workshop on The World Wide Web and Databases*, pages 172–183. Springer-Verlag.

Jesse Davis and Mark Goadrich. 2006. The relationship between precision-recall and roc curves. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, New York, NY, USA. ACM.

Xiaowen Ding, Bing Liu, and Philip S. Yu. 2008. A holistic lexicon-based approach to opinion mining. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 231–240. ACM.

Howard Ehrlichman and Rosalind Eichenstein. 1992. Private wishes: Gender similarities and difference. *Sex Roles*, 26(9):399–422.

Thomas Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl. 1):5228–5235.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of KDD '04, the ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM Press.

Thorsten Joachims. 1999. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Laura A. King and Sheri J. Broyles. 1997. Wishes, gender, personality, and well-being. *Journal of Personality*, 65(1):49–76.

Moshe Koppel and Itai Shtrimberg. 2004. Good news or bad news? let the market decide. In *AAAI Spring Symposium on Exploring Attitude and Affect in Text*, pages 86–88.

Saisuresh Krishnakumaran and Xiaojin Zhu. 2007. Hunting elusive metaphors using lexical resources. In *Proceedings of the Workshop on Computational Approaches to Figurative Language*, pages 13–20, Rochester, New York, April. Association for Computational Linguistics.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Rada Mihalcea and Hugo Liu. 2006. A corpus-based approach to finding happiness. In *Proceedings of AAAI-CAAW-06, the Spring Symposia on Computational Approaches to Analyzing Weblogs*.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Empirical Methods in Natural Language Processing*.

Norman A. Milgram and Wolfgang W. Riedel. 1969. Developmental and experiential factors in making wishes. *Child Development*, 40(3):763–771.

Gilad Mishne, Krisztian Balog, Maarten de Rijke, and Breyten Ernsting. 2007. Moodviews: Tracking and searching mood-annotated blog posts. In *Proceedings International Conf. on Weblogs and Social Media (ICWSM-2007)*, pages 323–324.

Tony Mullen and Robert Malouf. 2008. Taking sides: User classification for informal online political discourse. *Internet Research*, 18:177–190.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Fifth Conference on Applied Natural Language Processing*.

James Shanahan, Yan Qu, and Janyce Wiebe, editors. 2005. *Computing attitude and affect in text*. Springer, Dordrecht, The Netherlands.

George S. Speer. 1939. Oral and written wishes of rural and city school children. *Child Development*, 10(3):151–155.

G. K. Zipf. 1932. *Selected Studies of the Principle of Relative Frequency in Language*. Harvard University Press.

Predicting Risk from Financial Reports with Regression

Shimon Kogan
McCombs School of Business
University of Texas at Austin
Austin, TX 78712, USA
shimon.kogan@mccombs.utexas.edu

Dimitry Levin
Mellon College of Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dimitrylevin@gmail.com

Bryan R. Routledge
Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA 15213, USA
routledge@cmu.edu

Jacob S. Sagi
Owen Graduate School of Management
Vanderbilt University
Nashville, TN 37203, USA
Jacob.Sagi@Owen.Vanderbilt.edu

Noah A. Smith
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
nasmith@cs.cmu.edu

Abstract

We address a text regression problem: given a piece of text, predict a real-world continuous quantity associated with the text's meaning. In this work, the text is an SEC-mandated financial report published annually by a publicly-traded company, and the quantity to be predicted is *volatility* of stock returns, an empirical measure of financial risk. We apply well-known regression techniques to a large corpus of freely available financial reports, constructing regression models of volatility for the period following a report. Our models rival past volatility (a strong baseline) in predicting the target variable, and a single model that uses both can significantly outperform past volatility. Interestingly, our approach is more accurate for reports after the passage of the Sarbanes-Oxley Act of 2002, giving some evidence for the success of that legislation in making financial reports more informative.

1 Introduction

We consider a text regression problem: given a piece of text, predict a $\mathbb{R}$-valued quantity associated with that text. Specifically, we use a company's annual financial report to predict the financial risk of investment in that company, as measured empirically by a quantity known as *stock return volatility*.

Predicting financial risk is of clear interest to anyone who invests money in stocks and central to modern portfolio choice. Financial reports are a government-mandated artifact of the financial world that—one might hypothesize—contain a large amount of information about companies and their value. Indeed, it is an important question whether mandated disclosures are informative, since they are meant to protect investors but are costly to produce.

The intrinsic properties of the problem are attractive as a test-bed for NLP research. First, there is no controversy about the usefulness or existential reality of the output variable (volatility). Statistical NLP often deals in the prediction of variables ranging from text categories to linguistic structures to novel utterances. While many of these targets are uncontroversially useful, they often suffer from evaluation difficulties and disagreement among annotators. The output variable in this work is a statistic summarizing facts about the real world; it is not subject to any kind of human expertise, knowledge, or intuition. Hence this prediction task provides a new, objective test-bed for any kind of linguistic analysis.

Second, many NLP problems rely on costly annotated resources (e.g., treebanks or aligned bilingual corpora). Because the text and historical financial data used in this work are freely available (by law) and are generated as a by-product of the American

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 272–280,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

economy, old and new data can be obtained by anyone with relatively little effort.

In this paper, we demonstrate that predicting financial volatility automatically from a financial report is a novel, challenging, and easily evaluated natural language understanding task. We show that a very simple representation of the text (essentially, bags of unigrams and bigrams) can rival and, in combination, improve over a strong baseline that does not use the text. Analysis of the learned models provides insights about what can make this problem more or less difficult, and suggests that disclosure-related legislation led to more transparent reporting.

2 Stock Return Volatility

Volatility is often used in finance as a measure of *risk*. It is measured as the standard deviation of a stock's returns over a finite period of time. A stock will have high volatility when its price fluctuates widely and low volatility when its price remains more or less constant.

Let $r_t = \frac{P_t}{P_{t-1}} - 1$ be the return on a given stock between the close of trading day $t-1$ and day t, where P_t is the (dividend-adjusted) closing stock price at date t. The measured volatility over the time period from day $t-\tau$ to day t is equal to the sample s.d.:

$$v_{[t-\tau,t]} = \sqrt{\sum_{i=0}^{\tau} (r_{t-i} - \bar{r})^2 \Big/ \tau} \qquad (1)$$

where $\bar{r}$ is the sample mean of r_t over the period. In this work, the above estimate will be treated as the true output variable on training and testing data.

It is important to note that predicting volatility is not the same as predicting *returns* or *value*. Rather than trying to predict how well a stock will perform, we are trying to predict how stable its price will be over a future time period. It is, by now, received wisdom in the field of economics that predicting a stock's *performance*, based on easily accessible public information, is difficult. This is an attribute of well-functioning (or "efficient") markets and a cornerstone of the so-called "efficient market hypothesis" (Fama, 1970). By contrast, the idea that one can predict a stock's level of *risk* using public information is uncontroversial and a basic assumption made by many economically sound pricing models. A large body of research in finance suggests that the two types of quantities are very different: while predictability of returns could be easily traded away by the virtue of buying/selling stocks that are under- or over-valued (Fama, 1970), similar trades are much more costly to implement with respect to predictability of volatility (Dumas et al., 2007). By focusing on volatility prediction, we avoid taking a stance on whether or not the United States stock market is informationally efficient.

3 Problem Formulation

Given a text document $\mathbf{d}$, we seek to predict the value of a continuous variable v. We do this via a parameterized function f:

$$\hat{v} = f(\mathbf{d}; \mathbf{w}) \qquad (2)$$

where $\mathbf{w} \in \mathbb{R}^d$ are the parameters or weights. Our approach is to learn a human-interpretable $\mathbf{w}$ from a collection of N training examples $\{\langle \mathbf{d}_i, v_i \rangle\}_{i=1}^N$, where each $\mathbf{d}_i$ is a document and each $v_i \in \mathbb{R}$.

Support vector regression (Drucker et al., 1997) is a well-known method for training a regression model. SVR is trained by solving the following optimization problem:

$$\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N} \sum_{i=1}^{N} \underbrace{\max\left(0, \left|v_i - f(\mathbf{d}_i; \mathbf{w})\right| - \epsilon\right)}_{\epsilon\text{-insensitive loss function}} \qquad (3)$$

where C is a regularization constant and ϵ controls the training error.[1] The training algorithm finds weights $\mathbf{w}$ that define a function f minimizing the (regularized) empirical risk.

Let h be a function from documents into some vector-space representation $\subseteq \mathbb{R}^d$. In SVR, the function f takes the form:

$$f(\mathbf{d}; \mathbf{w}) \;=\; h(\mathbf{d})^\top \mathbf{w} \;=\; \sum_{i=1}^{N} \alpha_i K(\mathbf{d}, \mathbf{d}_i) \qquad (4)$$

where Equation 4 re-parameterizes f in terms of a kernel function K with "dual" weights α_i. K can

[1] Given the embedding h of documents in $\mathbb{R}^d$, ϵ defines a "slab" (region between two parallel hyperplanes, sometimes called the "ϵ-tube") in $\mathbb{R}^{d+1}$ through which each $\langle h(\mathbf{d}_i), f(\mathbf{d}_i; \mathbf{w}) \rangle$ must pass in order to have zero loss.

year	words	documents	words/doc.
1996	5.5M	1,408	3,893
1997	9.3M	2,260	4,132
1998	11.8M	2,462	4,808
1999	14.5M	2,524	5,743
2000	13.4M	2,425	5,541
2001	15.4M	2,596	5,928
2002	22.7M	2,846	7,983
2003	35.3M	3,612	9,780
2004	38.9M	3,559	10,936
2005	41.9M	3,474	12,065
2006	38.8M	3,308	11,736
total	247.7M	26,806	9,240

Table 1: Dimensions of the dataset used in this paper, after filtering and tokenization. The near doubling in average document size during 2002–3 is possibly due to the passage of the Sarbanes-Oxley Act of 2002 in the wake of Enron's accounting scandal (and numerous others).

be seen as a similarity function between two documents. At test time, a new example is compared to a subset of the training examples (those with $\alpha_i \neq 0$); typically with SVR this set is sparse. With the linear kernel, the primal and dual weights relate linearly:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i h(\mathbf{d}_i) \quad (5)$$

The full details of SVR and its implementation are beyond the scope of this paper; interested readers are referred to Schölkopf and Smola (2002). SVM^{light} (Joachims, 1999) is a freely available implementation of SVR training that we used in our experiments.[2]

4 Dataset

In the United States, the Securities Exchange Commission mandates that all publicly-traded corporations produce annual reports known as "Form 10-K." The report typically includes information about the history and organization of the company, equity and subsidiaries, as well as financial information. These reports are available to the public and published on the SEC's web site.[3] The structure of the 10-K is specified in detail in the legislation. We have collected 54,379 reports published over the period 1996–2006 from 10,492 different companies. Each report comes with a date of publication, which is important for tying the text back to the financial variables we seek to predict.

From the perspective of predicting future events, one section of the 10-K is of special interest: Section 7, known as "management's discussion and analysis of financial conditions and results of operations" (MD&A), and in particular Subsection 7A, "quantitative and qualitative disclosures about market risk." Because Section 7 is where the most important forward-looking content is most likely to be found, we filter other sections from the reports. The filtering is done automatically using a short, hand-written Perl script that seeks strings loosely matching the Section 7, 7A, and 8 headers, finds the longest reasonable "Section 7" match (in words) of more than 1,000 whitespace-delineated tokens.

Section 7 typically begins with an introduction like this (from ABC's 1998 Form 10-K, before tokenization for readability; boldface added):

> The following discussion and analysis of ABC's consolidated financial condition and consolidated results of operation should be read in conjunction with ABC's Consolidated Financial Statements and Notes thereto included elsewhere herein. This discussion contains certain **forward-looking statements which involve risks and uncertainties**. ABC's actual results could differ materially from the results expressed in, or implied by, such statements. See "Regarding Forward-Looking Statements."

Not all of the documents downloaded pass the filter at all, and for the present work we have only used documents that do pass the filter. (One reason for the failure of the filter is that many 10-K reports include Section 7 "by reference," so the text is not directly included in the document.)

In addition to the reports, we used the Center for Research in Security Prices (CRSP) US Stocks Database to obtain the price return series along with other firm characteristics.[4] We proceeded to calculate two volatilities for each firm/report observation: the twelve months prior to the report ($v^{(-12)}$) and the twelve months after the report ($v^{(+12)}$).

[2] Available at `http://svmlight.joachims.org`.

[3] `http://www.sec.gov/edgar.shtml`

[4] The text and volatility data are publicly available at `http://www.ark.cs.cmu.edu/10K`.

Tokenization was applied to the text, including punctuation removal, downcasing, collapsing all digit sequences,[5] and heuristic removal of remnant markup. Table 1 gives statistics on the corpora used in this research; this is a subset of the corpus for which there is no missing volatility information. The drastic increase in length during the 2002–2003 period might be explained by the passage by the US Congress of the Sarbanes-Oxley Act of 2002 (and related SEC and exchange rules), which imposed revised standards on reporting practices of publicly-traded companies in the US.

5 Baselines and Evaluation Method

Volatility displays an effect known as autoregressive conditional heteroscedasticity (Engle, 1982). This means that the variance in a stock's return tends to change gradually. Large changes in price are presaged by other changes, and periods of stability tend to continue. Volatility is, generally speaking, not constant, yet prior volatility (e.g., $v^{(-12)}$) is a very good predictor of future volatility (e.g., $v^{(+12)}$). At the granularity of a year, which we consider here because the 10-K reports are annual, there are no existing models of volatility that are widely agreed to be significantly more accurate than our historical volatility baseline. We tested a state-of-the-art model known as GARCH$(1,1)$ (Engle, 1982; Bollerslev, 1986) and found that it was no stronger than our historical volatility baseline on this sample.

Throughout this paper, we will report performance using the mean squared error between the predicted and true log-volatilities:[6]

$$\text{MSE} = \frac{1}{N'}\sum_{i=1}^{N'}\left(\log(v_i) - \log(\hat{v}_i)\right)^2 \qquad (6)$$

where N' is the size of the test set, given in Table 1.

6 Experiments

In our experiments, we vary h (the function that maps inputs to a vector space) and the subset of the data used for training. We will always report performance over test sets consisting of one year's worth of data (the subcorpora described in Table 1). In this work, we focus on predicting the volatility over the year following the report ($v^{(+12)}$). In all experiments, $\epsilon = 0.1$ and C is set using the default choice of SVM^{light}, which is the inverse of the average of $h(\mathbf{d})^\top h(\mathbf{d})$ over the training data.[7]

6.1 Feature Representation

We first consider how to represent the 10-K reports. We adopt various document representations, all using word features. Let M be the vocabulary size derived from the training data.[8] Let $\text{freq}(x_j; \mathbf{d})$ denote the number of occurrences of the jth word in the vocabulary in document $\mathbf{d}$.

- **TF**: $h_j(\mathbf{d}) = \frac{1}{|\mathbf{d}|}\text{freq}(x_j; \mathbf{d}), \forall j \in \{1, ..., M\}$.
- **TFIDF**: $h_j(\mathbf{d}) = \frac{1}{|\mathbf{d}|}\text{freq}(x_j; \mathbf{d}) \times \log(N/|\{\mathbf{d} : \text{freq}(x_j; \mathbf{d}) > 0\}|)$, where N is the number of documents in the training set. This is the classic "TFIDF" score.
- **LOG1P**: $h_j(\mathbf{d}) = \log(1 + \text{freq}(x_j; \mathbf{d}))$. Rather than normalizing word frequencies as for TF, this score dampens them with a logarithm. We also include a variant of LOG1P where terms are the union of unigrams and bigrams.

Note that each of these preserves sparsity; when $\text{freq}(x_j; \mathbf{d}) = 0$, $h_j(\mathbf{d}) = 0$ in all cases.

For interpretability of results, we use a linear kernel. The usual bias weight b is included. We found it convenient to work in the logarithmic domain for the predicted variable, predicting $\log v$ instead of v, since volatility is always nonnegative. In this setting, the predicted volatility takes the form:

$$\log \hat{v} \;=\; b + \sum_{j=1}^{M} w_j h_j(\mathbf{d}) \qquad (7)$$

Because the goal of this work is to explore how text might be used to predict volatility, we also wish

[5] While numerical information is surely informative about risk, recall that our goal is to find indicators of risk expressed in the text; automatic predictors of risk from numerical data would use financial data streams directly, not text reports.

[6] We work in the log domain because it is standard in finance, due to the dynamic range of actual volatilities; the distribution over $\log v$ across companies tends to have a bell shape.

[7] These values were selected after preliminary and cursory exploration with 1996–2000 as training data and 2001 as the test set. While the effects of ϵ and C were not large, further improvements may be possible with more careful tuning.

[8] Preliminary experiments that filtered common or rare words showed a negligible or deleterious effect on performance.

	features	2001	2002	2003	2004	2005	2006	micro-ave.
history	$v^{(-12)}$ (baseline)	0.1747	0.1600	0.1873	0.1442	0.1365	0.1463	0.1576
	$v^{(-12)}$ (SVR with bias)	0.2433	0.4323	**0.1869**	0.2717	0.3184	5.6778	1.2061
	$v^{(-12)}$ (SVR without bias)	0.2053	0.1653	0.2051	**0.1337**	0.1405	0.1517	0.1655
words	TF	0.2219	0.2571	0.2588	0.2134	0.1850	0.1862	0.2197
	TFIDF	0.2033	0.2118	0.2178	0.1660	0.1544	0.1599	0.1842
	LOG1P	0.2107	0.2214	0.2040	0.1693	0.1581	0.1715	0.1873
	LOG1P, bigrams	0.1968	0.2015	***0.1729**	0.1500	0.1394	0.1532	0.1667
both	TF+	0.1885	0.1616	0.1925	***0.1230**	***0.1272**	***0.1402**	***0.1541**
	TFIDF+	0.1919	0.1618	0.1965	***0.1246**	***0.1276**	***0.1403**	***0.1557**
	LOG1P+	0.1846	0.1764	***0.1671**	***0.1309**	***0.1319**	**0.1458**	***0.1542**
	LOG1P+, bigrams	0.1852	0.1792	***0.1599**	***0.1352**	***0.1307**	**0.1448**	***0.1538**

Table 2: MSE (Eq. 6) of different models on test data predictions. Lower values are better. Boldface denotes improvements over the baseline, and * denotes significance compared to the baseline under a permutation test ($p < 0.05$).

to see whether text adds information beyond what can be predicted using historical volatility alone (the baseline, $v^{(-12)}$). We therefore consider models augmented with an additional feature, defined as $h_{M+1} = \log v^{(-12)}$. Since this is historical information, it is always available when the 10-K report is published. These models are denoted TF+, TFIDF+, and LOG1P+.

The performance of these models, compared to the baseline from Section 5, is shown in Table 2. We used as training examples all reports from the five-year period preceding the test year (so six experiments on six different training and test sets are shown in the figure). We also trained SVR models on the single feature $v^{(-12)}$, with and without bias weights (b in Eq. 7); these are usually worse and never signficantly better than the baseline.

Strikingly, the models that use *only* the text to predict volatility come very close to the historical baseline in some years. That a text-only method (LOG1P with bigrams) for predicting future risk comes within 5% of the error of a strong baseline (2003–6) shows promise for the overall approach. A combined model improves substantially over the baseline in four out of six years (2003–6), and this difference is usually robust to the representation used. Table 3 shows the most strongly weighted terms in each of the text-only LOG1P models (including bigrams). These weights are recovered using the relationship expressed in Eq. 5.

6.2 Training Data Effects

It is well known that more training data tend to improve the performance of a statistical method; however, the standard assumption is that the training data are drawn from the same distribution as the test data. In this work, where we seek to predict the future based on data from past, that assumption is obviously violated. It is therefore an open question whether more data (i.e., looking farther into the past) is helpful for predicting volatility, or whether it is better to use only the most recent data.

Table 4 shows how performance varies when one, two, or five years of historical training data are used, averaged across test years. In most cases, using more training data (from a longer historical period) is helpful, but not always. One interesting trend, not shown in the aggregate statistics of Table 4, is that recency of the training set affected performance much more strongly in earlier train/test splits (2001–3) than later ones (2004–6). This experiment leads us to conclude that temporal changes in financial reporting make training data selection nontrivial. Changes in the macro economy and specific businesses make older reports less relevant for prediction. For example, regulatory changes like Sarbanes-Oxley, variations in the business cycle, and technological innovation like the Internet influence both the volatility and the 10-K text.

6.3 Effects of Sarbanes-Oxley

We noted earlier that the passage of the Sarbanes-Oxley Act of 2002, which sought to reform financial reporting, had a clear effect on the *lengths* of the 10-K reports in our collection. But are the reports more informative? This question is important, because producing reports is costly; we present an empirical argument based on our models that the legis-

1996–2000		1997–2001		1998–2002		1999–2003		2000–2004		2001–2005		high $\hat{v}$
net loss	0.026	year #	0.028	loss	0.023	loss	0.026	loss	0.025	loss	0.026	
year #	0.024	net loss	0.023	net loss	0.020	net loss	0.020	net loss	0.017	net loss	0.018	↑
loss	0.020	expenses	0.020	expenses	0.017	expenses	0.017	year #	0.016	going concern	0.014	
expenses	0.019	loss	0.020	year #	0.015	going concern	0.015	expenses	0.015	expenses	0.014	
covenants	0.017	experienced	0.015	obligations	0.015	year #	0.015	going concern	0.014	a going	0.014	
diluted	0.014	of $#	0.015	financing	0.014	financing	0.014	a going	0.013	personnel	0.013	
convertible	0.014	covenants	0.015	convertible	0.014	a going	0.014	administrative	0.013	financing	0.013	
date	0.014	additional	0.014	additional	0.014	additional	0.013	personnel	0.013	administrative	0.012	
longterm	-0.014	merger agreement	-0.015	unsecured	-0.012	distributions	-0.012	distributions	-0.011	policies	-0.011	
rates	-0.015	dividends	-0.015	earnings	-0.012	annual	-0.012	insurance	-0.011	by the	-0.011	
dividend	-0.015	unsecured	-0.017	distributions	-0.012	dividend	-0.012	critical accounting	-0.012	earnings	-0.011	
unsecured	-0.015	dividend	-0.017	dividends	-0.015	dividends	-0.012	lower interest	-0.012	dividends	-0.012	
merger agreement	-0.017	properties	-0.018	income	-0.016	rates	-0.013	dividends	-0.013	unsecured	-0.012	
properties	-0.018	net income	-0.019	properties	-0.016	properties	-0.015	properties	-0.014	properties	-0.013	
income	-0.021	income	-0.021	net income	-0.019	rate	-0.019	rate	-0.017	rate	-0.014	↓
rate	-0.022	rate	-0.025	rate	-0.022	net income	-0.023	net income	-0.021	net income	-0.018	low $\hat{v}$

Table 3: Most strongly-weighted terms in models learned from various time periods (LOG1P model with unigrams and bigrams). "#" denotes any digit sequence.

features	1	2	5
TF+	0.1509	**0.1450**	0.1541
TFIDF+	0.1512	**0.1455**	0.1557
LOG1P+	0.1621	0.1611	**0.1542**
LOG1P+, bigrams	0.1617	0.1588	**0.1538**

Table 4: MSE of volatility predictions using reports from varying historical windows (1, 2, and 5 years), micro-averaged across six train/test scenarios. Boldface marks best in a row. The historical baseline achieves 0.1576 MSE (see Table 2).

lation has actually been beneficial.

Our experimental results in Section 6.1, in which volatility in the years 2004–2006 was more accurately predicted from the text than in 2001–2002, suggest that the Sarbanes-Oxley Act led to more informative reports. We compared the learned weights (LOG1P+, unigrams) between the six overlapping five-year windows ending in 2000–2005; measured in L_1 distance, these were, in consecutive order, $\langle 52.2,\ 59.9,\ 60.7,\ 55.3,\ 52.3 \rangle$; the biggest differences came between 2001 and 2002 and between 2002 and 2003. (Firms are most likely to have begun compliance with the new law in 2003 or 2004.) The same pattern held when only words appearing in all five models were considered. Variation in the recency/training set size tradeoff (§6.2), particularly during 2002–3, also suggests that there were substantial changes in the reports during that time.

6.4 Qualitative Evaluation

One of the advantages of a linear model is that we can explore what each model discovers about different unigram and bigram terms. Some manually selected examples of terms whose learned weights (w) show interesting variation patterns over time are shown in Figure 1, alongside term frequency patterns, for the text-only LOG1P model (with bigrams). These examples were suggested by experts in finance from terms with weights that were both large and variable (across training sets).

A particularly interesting case, in light of Sarbanes-Oxley, is the term *accounting policies*. Sarbanes-Oxley mandated greater discussion of accounting policy in the 10-K MD&A section. Before 2002 this term indicates high volatility, perhaps due to complicated off-balance sheet transactions or unusual accounting policies. Starting in 2002, explicit mention of accounting policies indi-

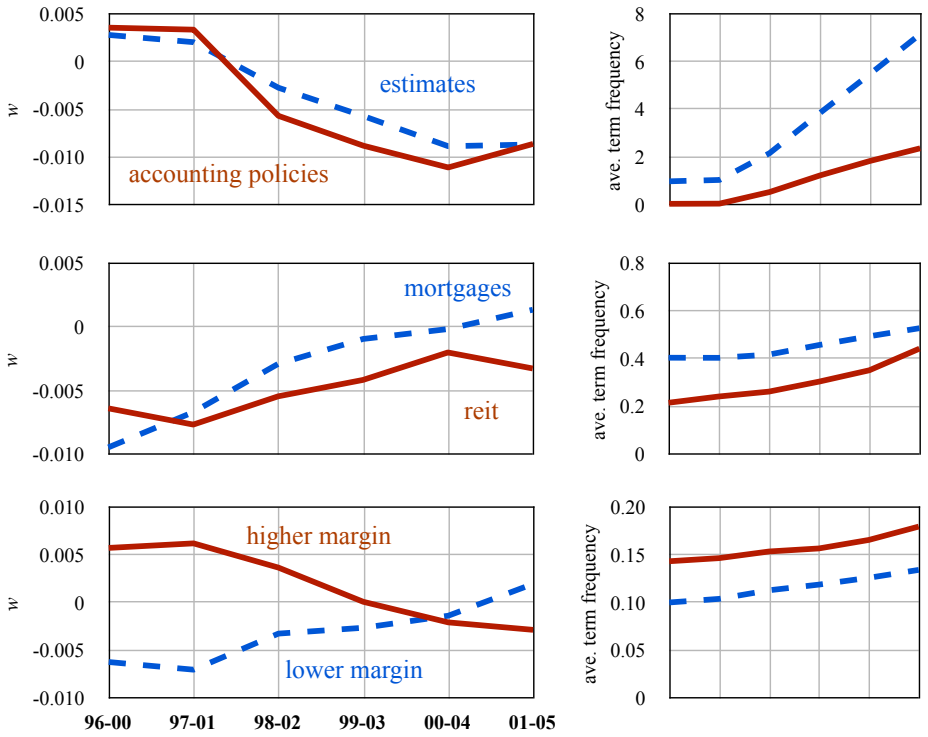


Figure 1: Left: learned weights for selected terms across models trained on data from different time periods (x-axis). These weights are from the LOG1P (unigrams and bigrams) models trained on five-year periods, the same models whose extreme weights are summarized in Tab. 3. Note that all weights are within 0 ± 0.026. Right: the terms' average frequencies (by document) over the same periods.

cates lower volatility. The frequency of the term also increases drastically over the same period, suggesting that the earlier weights may have been inflated. A more striking example is *estimates*, which averages one occurrence per document even in the 1996–2000 period, experiences the same term frequency explosion, and goes through a similar weight change, from strongly indicating high volatility to strongly indicating low volatility.

As a second example, consider the terms *mortgages* and *reit* (Real Estate Investment Trust, a tax designation for businesses that invest in real estate). Given the importance of the housing and mortgage market over the past few years, it is interesting to note that the weight on both of these terms increases over the period from a strong low volatility term to a weak indicator of high volatility. It will be interesting to see how the dramatic decline in housing prices in late 2007, and the fallout created in credit markets in 2008, is reflected in future models.

Finally, notice that *high margin* and *low margin*, whose frequency patterns are fairly flat "switch places," over the sample: first indicating high and low volatility, respectively, then low and high. There is no *a priori* reason to expect high or low margins would be associated with high or low stock volatility. However, this is an interesting example where bigrams are helpful (the word margin by itself is uninformative) and indicates that predicting risk is highly time-dependent.

6.5 Delisting

An interesting but relatively infrequent phenomenon is the **delisting** of a company, i.e., when it ceases to be traded on a particular exchange due to dissolution after bankruptcy, a merger, or violation of exchange rules. The relationship between volatility and delisting has been studied by Merton (1974), among others. Our dataset includes a small number of cases where the volatility figures for the period following the publication of a 10-K report are unavailable because the company was delisted. Learning to predict delisting is extremely difficult because fewer than 4% of the 2001–6 10-K reports precede delisting.

Using the LOG1P representation, we built a linear SVM classifier for each year in 2001–6 (trained on the five preceding years' data) to predict whether a company will be delisted following its 10-K report. Performance for various precision measures is shown in Table 5. Notably, for 2001–4 we achieve

precision (%) at ...	'01	'02	'03	'04	'05	'06
recall = 10%	80	93	79	100	47	21
n = 5	100	100	40	100	60	80
n = 10	80	90	70	90	60	70
n = 100	38	48	53	29	24	20
oracle F_1 (%)	35	42	44	36	31	16

6	bulletin, creditors, dip, otc
5	court
4	chapter, debtors, filing, prepetition
3	bankruptcy
2	concern, confirmation, going, liquidation
1	debtorinpossession, delisted, nasdaq, petition

Table 5: Left: precision of delisting predictions. The "oracle F_1" row shows the maximal F_1 score obtained for any n. Right: Words most strongly predicting delisting of a company. The number is how many of the six years (2001–6) the word is among the ten most strongly weighted. There were no clear patterns across years for words predicting that a company would *not* be delisted. The word *otc* refers to "over-the-counter" trading, a high-risk market.

above 75% precision at 10% recall. Our best (oracle) F_1 scores occur in 2002 and 2003, suggesting again a difference in reports around Sarbanes-Oxley. Table 5 shows words associated with delisting.

7 Related Work

In NLP, regression is not widely used, since most natural language-related data are discrete. Regression methods were pioneered by Yang and Chute (1992) and Yang and Chute (1993) for information retrieval purposes, but the predicted continuous variable was not an end in itself in that work. Blei and McAuliffe (2007) used latent "topic" variables to predict movie reviews and popularity from text. Lavrenko et al. (2000b) and Lavrenko et al. (2000a) modeled influences between text and time series financial data (stock prices) using language models. Farther afield, Albrecht and Hwa (2007) used SVR to train machine translation evaluation metrics to match human evaluation scores and compared techniques using correlation. Regression has also been used to order sentences in extractive summarization (Biadsy et al., 2008).

While much of the information relevant for investors is communicated through text (rather than numbers), only recently is this link explored. Some papers relate news articles to earning forecasts, stock returns, volatility, and volume (Koppel and Shtrimberg, 2004; Tetlock, 2007; Tetlock et al., 2008; Gaa, 2007; Engelberg, 2007). Das and Chen (2001) and Antweiler and Frank (2004) ask whether messages posted on message boards can help explain stock performance, while Li (2005) measures the association between frequency of words associated with risk and subsequent stock returns. Weiss-Hanley and Hoberg (2008) study initial public offering disclosures using word statistics. Many researchers have focused the related problem of predicting sentiment and opinion in text (Pang et al., 2002; Wiebe and Riloff, 2005), sometimes connected to extrinsic values like prediction markets (Lerman et al., 2008).

In contrast to text regression, text *classification* comprises a widely studied set of problems involving the prediction of *categorial* variables related to text. Applications have included the categorization of documents by topic (Joachims, 1998), language (Cavnar and Trenkle, 1994), genre (Karlgren and Cutting, 1994), author (Bosch and Smith, 1998), sentiment (Pang et al., 2002), and desirability (Sahami et al., 1998). Text categorization has served as a test application for nearly every machine learning technique for discrete classification.

8 Conclusion

We have introduced and motivated a new kind of task for NLP: *text regression*, in which text is used to make predictions about measurable phenomena in the real world. We applied the technique to predicting financial volatility from companies' 10-K reports, and found text regression model predictions to correlate with true volatility nearly as well as historical volatility, and a combined model to perform even better. Further, improvements in accuracy and changes in models after the passage of the Sarbanes-Oxley Act suggest that financial reporting reform has had interesting and measurable effects.

Acknowledgments

The authors are grateful to Jamie Callan, Chester Spatt, Anthony Tomasic, Yiming Yang, and Stanley Zin for helpful discussions, and to the anonymous reviewers for useful feedback. This research was supported by grants from the Institute for Quantitative Research in Finanace and from the Center for Analytical Research in Technology at the Tepper School of Business, Carnegie Mellon University.

References

J. S. Albrecht and R. Hwa. 2007. Regression for sentence-level MT evaluation with pseudo references. In *Proc. of ACL*.

W. Antweiler and M. Z. Frank. 2004. Is all that talk just noise? the information content of internet stock message boards. *Journal of Finance*, 59:1259–1294.

F. Biadsy, J. Hirschberg, and E. Filatova. 2008. An unsupervised approach to biography production using Wikipedia. In *Proc. of ACL*.

D. M. Blei and J. D. McAuliffe. 2007. Supervised topic models. In *Advances in NIPS 21*.

T. Bollerslev. 1986. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31:307–327.

R. Bosch and J. Smith. 1998. Separating hyperplanes and the authorship of the disputed Federalist papers. *American Mathematical Monthly*, 105(7):601–608.

W. B. Cavnar and J. M. Trenkle. 1994. n-gram-based text categorization. In *Proc. of SDAIR*.

S. Das and M. Chen. 2001. Yahoo for Amazon: Extracting market sentiment from stock mesage boards. In *Proc. of Asia Pacific Finance Association Annual Conference*.

H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik. 1997. Support vector regression machines. In *Advances in NIPS 9*.

B. Dumas, A. Kurshev, and R. Uppal. 2007. Equilibrium portfolio strategies in the presence of sentiment risk and excess volatility. Swiss Finance Institute Research Paper No. 07-37.

J. Engelberg. 2007. Costly information processing: Evidence from earnings announcements. Working paper, Northwestern University.

R. F. Engle. 1982. Autoregressive conditional heteroscedasticity with estimates of variance of united kingdom inflation. *Econometrica*, 50:987–1008.

E. F. Fama. 1970. Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, 25(2):383–417.

C. Gaa. 2007. Media coverage, investor inattention, and the market's reaction to news. Working paper, University of British Columbia.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of ECML*.

T. Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

J. Karlgren and D. Cutting. 1994. Recognizing text genres with simple metrics using discriminant analysis. In *Proc. of COLING*.

M. Koppel and I. Shtrimberg. 2004. Good news or bad news? let the market decide. In *AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*.

V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan. 2000a. Language models for financial news recommendation. In *Proc. of CIKM*.

V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan. 2000b. Mining of concurrent text and time series. In *Proc. of KDD*.

K. Lerman, A. Gilder, M. Dredze, and F. Pereira. 2008. Reading the markets: Forecasting public opinion of political candidates by news analysis. In *COLING*.

F. Li. 2005. Do stock market investors understand the risk sentiment of corporate annual reports? Working Paper, University of Michigan.

R. Merton. 1974. On the pricing of corporate debt: The risk structure of interest rates. *Journal of Finance*, 29:449–470.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proc. of EMNLP*.

M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. 1998. A Bayesian approach to filtering junk email. In *Proc. of AAAI Workshop on Learning for Text Categorization*.

B. Schölkopf and A. J. Smola. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

P. C. Tetlock, M. Saar-Tsechansky, and S. Macskassy. 2008. More than words: Quantifying language to measure firms' fundamentals. *Journal of Finance*, 63(3):1437–1467.

P. C. Tetlock. 2007. Giving content to investor sentiment: The role of media in the stock market. *Journal of Finance*, 62(3):1139–1168.

K. Weiss-Hanley and G. Hoberg. 2008. Strategic disclosure and the pricing of initial public offerings. Working paper.

J. Wiebe and E. Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *CICLing*.

Y. Yang and C. G. Chute. 1992. A linear least squares fit mapping method for information retrieval from natural language texts. In *Proc. of COLING*.

Y. Yang and C. G. Chute. 1993. An application of least squares fit mapping to text information retrieval. In *Proc. of SIGIR*.

Domain Adaptation with Latent Semantic Association for Named Entity Recognition

Honglei Guo Huijia Zhu Zhili Guo Xiaoxun Zhang Xian Wu and Zhong Su
IBM China Research Laboratory
Beijing, P. R. China
{guohl, zhuhuiji, guozhili, zhangxx, wuxian, suzhong}@cn.ibm.com

Abstract

Domain adaptation is an important problem in named entity recognition (NER). NER classifiers usually lose accuracy in the domain transfer due to the different data distribution between the source and the target domains. The major reason for performance degrading is that each entity type often has lots of domain-specific term representations in the different domains. The existing approaches usually need an amount of labeled target domain data for tuning the original model. However, it is a labor-intensive and time-consuming task to build annotated training data set for every target domain. We present a domain adaptation method with latent semantic association (LaSA). This method effectively overcomes the data distribution difference without leveraging any labeled target domain data. LaSA model is constructed to capture latent semantic association among words from the unlabeled corpus. It groups words into a set of concepts according to the related context snippets. In the domain transfer, the original term spaces of both domains are projected to a concept space using LaSA model at first, then the original NER model is tuned based on the semantic association features. Experimental results on English and Chinese corpus show that LaSA-based domain adaptation significantly enhances the performance of NER.

1 Introduction

Named entities (NE) are phrases that contain names of persons, organizations, locations, etc. NER is an important task in information extraction and natural language processing (NLP) applications. Supervised learning methods can effectively solve NER problem by learning a model from manually labeled data (Borthwick, 1999; Sang and Meulder, 2003; Gao et al., 2005; Florian et al., 2003). However, empirical study shows that NE types have different distribution across domains (Guo et al., 2006). Trained NER classifiers in the source domain usually lose accuracy in a new target domain when the data distribution is different between both domains.

Domain adaptation is a challenge for NER and other NLP applications. In the domain transfer, the reason for accuracy loss is that each NE type often has various specific term representations and context clues in the different domains. For example, {“economist”, “singer”, “dancer”, “athlete”, “player”, “philosopher”, ...} are used as context clues for NER. However, the distribution of these representations are varied with domains. We expect to do better domain adaptation for NER by exploiting latent semantic association among words from different domains. Some approaches have been proposed to group words into “topics” to capture important relationships between words, such as Latent Semantic Indexing (LSI) (Deerwester et al., 1990), probabilistic Latent Semantic Indexing (pLSI) (Hofmann, 1999), Latent Dirichlet Allocation (LDA) (Blei et al., 2003). These models have been successfully employed in topic modeling, dimensionality reduction for text categorization (Blei et al., 2003), ad hoc IR (Wei and Croft., 2006), and so on.

In this paper, we present a domain adaptation method with latent semantic association. We focus

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 281–289,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

on capturing the hidden semantic association among words in the domain adaptation. We introduce the LaSA model to overcome the distribution difference between the source domain and the target domain. LaSA model is constructed from the unlabeled corpus at first. It learns latent semantic association among words from their related context snippets. In the domain transfer, words in the corpus are associated with a low-dimension concept space using LaSA model, then the original NER model is tuned using these generated semantic association features. The intuition behind our method is that words in one concept set will have similar semantic features or latent semantic association, and share syntactic and semantic context in the corpus. They can be considered as behaving in the same way for discriminative learning in the source and target domains. The proposed method associates words from different domains on a semantic level rather than by lexical occurrence. It can better bridge the domain distribution gap without any labeled target domain samples. Experimental results on English and Chinese corpus show that LaSA-based adaptation significantly enhances NER performance across domains.

The rest of this paper is organized as follows. Section 2 briefly describes the related works. Section 3 presents a domain adaptation method based on latent semantic association. Section 4 illustrates how to learn LaSA model from the unlabeled corpus. Section 5 shows experimental results on large-scale English and Chinese corpus across domains, respectively. The conclusion is given in Section 6.

2 Related Works

Some domain adaptation techniques have been employed in NLP in recent years. Some of them focus on quantifying the generalizability of certain features across domains. Roark and Bacchiani (2003) use maximum a posteriori (MAP) estimation to combine training data from the source and target domains. Chelba and Acero (2004) use the parameters of the source domain maximum entropy classifier as the means of a Gaussian prior when training a new model on the target data. Daume III and Marcu (2006) use an empirical Bayes model to estimate a latent variable model grouping instances into domain-specific or common across both domains. Daume III (2007) further augments the feature space on the instances of both domains. Jiang and Zhai (2006) exploit the domain structure contained in the training examples to avoid over-fitting the training domains. Arnold et al. (2008) exploit feature hierarchy for transfer learning in NER. Instance weighting (Jiang and Zhai, 2007) and active learning (Chan and Ng, 2007) are also employed in domain adaptation. Most of these approaches need the labeled target domain samples for the model estimation in the domain transfer. Obviously, they require much efforts for labeling the target domain samples.

Some approaches exploit the common structure of related problems. Ando et al. (2005) learn predicative structures from multiple tasks and unlabeled data. Blitzer et al. (2006, 2007) employ structural corresponding learning (SCL) to infer a good feature representation from unlabeled source and target data sets in the domain transfer. We present LaSA model to overcome the data gap across domains by capturing latent semantic association among words from unlabeled source and target data.

In addition, Miller et al. (2004) and Freitag (2004) employ distributional and hierarchical clustering methods to improve the performance of NER within a single domain. Li and McCallum (2005) present a semi-supervised sequence modeling with syntactic topic models. In this paper, we focus on capturing hidden semantic association among words in the domain adaptation.

3 Domain Adaptation Based on Latent Semantic Association

The challenge in domain adaptation is how to capture latent semantic association from the source and target domain data. We present a LaSA-based domain adaptation method in this section.

NER can be considered as a classification problem. Let X be a feature space to represent the observed word instances, and let Y be the set of class labels. Let $p_s(x, y)$ and $p_t(x, y)$ be the true underlying distributions for the source and the target domains, respectively. In order to minimize the efforts required in the domain transfer, we often expect to use $p_s(x, y)$ to approximate $p_t(x, y)$.

However, data distribution are often varied with the domains. For example, in the economics-to-

entertainment domain transfer, although many NE triggers (e.g. "company" and "Mr.") are used in both domains, some are totally new, like "dancer", "singer". Moreover, many useful words (e.g. "economist") in the economics NER are useless in the entertainment domain. The above examples show that features could change behavior across domains. Some useful predictive features from one domain are not predictive or do not appear in another domain. Although some triggers (e.g. "singer", "economist") are completely distinct for each domain, they often appear in the similar syntactic and semantic context. For example, triggers of person entity often appear as the subject of "visited", "said", etc, or are modified by "excellent", "popular", "famous" etc. Such latent semantic association among words provides useful hints for overcoming the data distribution gap of both domains.

Hence, we present a LaSA model $\theta_{s,t}$ to capture latent semantic association among words in the domain adaptation. $\theta_{s,t}$ is learned from the unlabeled source and target domain data. Each instance is characterized by its co-occurred context distribution in the learning. Semantic association feature in $\theta_{s,t}$ is a hidden random variable that is inferred from data. In the domain adaptation, we transfer the problem of semantic association mapping to a posterior inference task using LaSA model. Latent semantic concept association set of a word instance x (denoted by $SA(x)$) is generated by $\theta_{s,t}$. Instances in the same concept set are considered as behaving in the same way for discriminative learning in both domains. Even though word instances do not appear in a training corpus (or appear rarely) but are in similar context, they still might have relatively high probability in the same semantic concept set. Obviously, $SA(x)$ can better bridge the gap between the two distributions $p_s(y|x)$ and $p_t(y|x)$. Hence, LaSA model can enhance the estimate of the source domain distribution $p_s(y|x;\theta_{s,t})$ to better approximate the target domain distribution $p_t(y|x;\theta_{s,t})$.

4 Learning LaSA Model from Virtual Context Documents

In the domain adaptation, LaSA model is employed to find the latent semantic association structures of "words" in a text corpus. We will illustrate how to build LaSA model from words and their context snippets in this section. LaSA model actually can be considered as a general probabilistic topic model. It can be learned on the unlabeled corpus using the popular hidden topic models such as LDA or pLSI.

4.1 Virtual Context Document

The distribution of content words (e.g. nouns, adjectives) is usually varied with domains. Hence, in the domain adaptation, we focus on capturing the latent semantic association among content words. In order to learn latent relationships among words from the unlabeled corpus, each content word is characterized by a virtual context document as follows.

Given a content word x_i, the virtual context document of x_i (denoted by vd_{x_i}) consists of all the context units around x_i in the corpus. Let n be the total number of the sentences which contain x_i in the corpus. vd_{x_i} is constructed as follows.

$$vd_{x_i} = \{F(x_i^{s_1}), ..., F(x_i^{s_k}), ..., F(x_i^{s_n})\}$$

where, $F(x_i^{s_k})$ denotes the context feature set of x_i in the sentence s_k, $1 \leq k \leq n$.

Given the context window size {-t, t} (i.e. previous t words and next t words around x_i in s_k). $F(x_i^{s_k})$ usually consists of the following features.

1. Anchor unit $A_C^{x_i}$: the current focused word unit x_i.
2. Left adjacent unit $A_L^{x_i}$: The nearest left adjacent unit x_{i-1} around x_i, denoted by $A_L(x_{i-1})$.
3. Right adjacent unit $A_R^{x_i}$: The nearest right adjacent unit x_{i+1} around x_i, denoted by $A_R(x_{i+1})$.
4. Left context set $C_L^{x_i}$: the other left adjacent units $\{x_{i-t}, ..., x_{i-j}, ..., x_{i-2}\}$ ($2 \leq j \leq t$) around x_i, denoted by $\{C_L(x_{i-t}), ..., C_L(x_{i-j}), ..., C_L(x_{i-2})\}$.
5. Right context set $C_R^{x_i}$: the other right adjacent units $\{x_{i+2}, ..., x_{i+j}, ..., x_{i+t}\}$ ($2 \leq j \leq t$) around x_i, denoted by $\{C_R(x_{i+2}), ..., C_R(x_{i+j}), ..., C_R(x_{i+t})\}$.

For example, given x_i="*singer*", s_k="*This popular new singer attended the new year party*". Let the context window size be {-3,3}. $F(singer)$ = $\{singer, A_L(new), A_R(attend(ed)), C_L(this), C_L(popular), C_R(the), C_R(new)\}$.

vd_{x_i} actually describes the semantic and syntactic feature distribution of x_i in the domains. We construct the feature vector of x_i with all the observed context features in vd_{x_i}. Given vd_{x_i} =

$\{f_1, ..., f_j, ..., f_m\}$, f_j denotes jth context feature around x_i, $1 \leq j \leq m$, m denotes the total number of features in vd_{x_i}. The value of f_j is calculated by Mutual Information (Church and Hanks, 1990) between x_i and f_j.

$$Weight(f_j, x_i) = \log_2 \frac{P(f_j, x_i)}{P(f_j)P(x_i)} \quad (1)$$

where, $P(f_j, x_i)$ is the joint probability of x_i and f_j co-occurred in the corpus, $P(f_j)$ is the probability of f_j occurred in the corpus. $P(x_i)$ is the probability of x_i occurred in the corpus.

4.2 Learning LaSA Model

Topic models are statistical models of text that posit a hidden space of topics in which the corpus is embedded (Blei et al., 2003). LDA (Blei et al., 2003) is a probabilistic model that can be used to model and discover underlying topic structures of documents. LDA assumes that there are K "topics", multinomial distributions over words, which describes a collection. Each document exhibits multiple topics, and each word in each document is associated with one of them. LDA imposes a Dirichlet distribution on the topic mixture weights corresponding to the documents in the corpus. The topics derived by LDA seem to possess semantic coherence. Those words with similar semantics are likely to occur in the same topic. Since the number of LDA model parameters depends only on the number of topic mixtures and vocabulary size, LDA is less prone to over-fitting and is capable of estimating the probability of unobserved test documents. LDA is already successfully applied to enhance document representations in text classification (Blei et al., 2003), information retrieval (Wei and Croft., 2006).

In the following, we illustrate how to construct LDA-style LaSA model $\theta_{s,t}$ on the virtual context documents. Algorithm 1 describes LaSA model training method in detail, where, Function $AddTo(data, Set)$ denotes that $data$ is added to Set. Given a large-scale unlabeled data set D_u which consists of the source and target domain data, virtual context document for each candidate content word is extracted from D_u at first, then the value of each feature in a virtual context document is calculated using its Mutual Information (see Equation 1 in Section 4.1) instead of the counts when running

Algorithm 1: LaSA Model Training

```
Inputs:
• Unlabeled data set: D_u;
Outputs:
• LaSA model: θ_{s,t};
Initialization:
• Virtual context document set: VD_{s,t} = ∅;
• Candidate content word set: X_{s,t} = ∅;
Steps:
begin
    foreach content word x_i ∈ D_u do
        if Frequency(x_i) ≥ the predefined threshold then
            AddTo(x_i, X_{s,t});
    foreach x_k ∈ X_{s,t} do
        foreach sentence S_i ∈ D_u do
            if x_k ∈ S_i then
                F(x_k^{S_i}) ⟵
                {x_k, A_L^{x_k}, A_R^{x_k}, C_L^{x_k}, C_R^{x_k}};
                AddTo(F(x_k^{S_i}), vd_{x_k});
        AddTo(vd_{x_k}, VD_{s,t});
    • Generate LaSA model θ_{s,t} with Dirichlet distribution on VD_{s,t}.
end
```

LDA. LaSA model $\theta_{s,t}$ with Dirichlet distribution is generated on the virtual context document set $VD_{s,t}$ using the algorithm presented by Blei et al (2003).

1	2	3	4	5
customer	theater	company	Beijing	music
president	showplace	government	Hongkong	film
singer	courtyard	university	China	arts
manager	center	community	Japan	concert
economist	city	team	Singapore	party
policeman	gymnasium	enterprise	New York	Ballet
reporter	airport	bank	Vienna	dance
director	square	market	America	song
consumer	park	organization	Korea	band
dancer	building	agency	international	opera

Table 1: Top 10 nouns from 5 randomly selected topics computed on the economics and entertainment domains

LaSA model learns the posterior distribution to decompose words and their corresponding virtual context documents into topics. Table 1 lists top 10 nouns from a random selection of 5 topics computed on the unlabeled economics and entertainment domain data. As shown, words in the same topic are representative nouns. They actually are grouped into broad concept sets. For example, set 1, 3 and 4 correspond to nominal person, nominal organization and location, respectively. With a large-scale unlabeled corpus, we will have enough words assigned to each topic concept to better approximate the underlying semantic association distribution.

In LDA-style LaSA model, the topic mixture is drawn from a conjugate Dirichlet prior that remains the same for all the virtual context docu-

ments. Hence, given a word x_i in the corpus, we may perform posterior inference to determine the conditional distribution of the hidden topic feature variables associated with x_i. Latent semantic association set of x_i (denoted by $SA(x_i)$) is generated using Algorithm 2. Here, Multinomial($\theta_{s,t}(vd_{x_i})$) refers to sample from the posterior distribution over topics given a virtual document vd_{x_i}. In the domain adaptation, we do semantic association inference on the source domain training data using LaSA model at first, then the original source domain NER model is tuned on the source domain training data set by incorporating these generated semantic association features.

Algorithm 2: Generate Latent Semantic Association Set of Word x_i Using K-topic LaSA Model

Inputs:
- $\theta_{s,t}$: LaSA model with multinomial distribution;
- $Dirichlet(\alpha)$: Dirichlet distribution with parameter α;
- x_i: Content word;

Outputs:
- $SA(x_i)$: Latent semantic association set of x_i;

Steps:

begin
- Extract vd_{x_i} from the corpus.
- Draw topic weights $\theta_{s,t}(vd_{x_i})$ from $Dirichlet(\alpha)$;
- **foreach** f_j *in* vd_{x_i} **do**
  - draw a topic $z_j \in \{1,...,K\}$ from Multinomial($\theta_{s,t}(vd_{x_i})$);
  - $AddTo(z_j, Topics(vd_{x_i}))$;
- Rank all the topics in $Topics(vd_{x_i})$;
- $SA(x_i) \longleftarrow$ top n topics in $Topics(vd_{x_i})$;

end

LaSA model better models latent semantic association distribution in the source and the target domains. By grouping words into concepts, we effectively overcome the data distribution difference of both domains. Thus, we may reduce the number of parameters required to model the target domain data, and improve the quality of the estimated parameters in the domain transfer. LaSA model extends the traditional bag-of-words topic models to context-dependence concept association model. It has potential use for concept grouping.

5 Experiments

We evaluate LaSA-based domain adaptation method on both English and Chinese corpus in this section. In the experiments, we focus on recognizing person (PER), location (LOC) and organization (ORG) in the given four domains, including economics (Eco), entertainment (Ent), politics (Pol) and sports (Spo).

5.1 Experimental setting

In the NER domain adaptation, nouns and adjectives make a significant impact on the performance. Thus, we focus on capturing latent semantic association for high-frequency nouns and adjectives (i.e. occurrence count $\geq$ 50) in the unlabeled corpus. LaSA models for nouns and adjectives are learned from the unlabeled corpus using Algorithm 1 (see section 4.2), respectively. Our empirical study shows that better adaptation is obtained with a 50-topic LaSA model. Therefore, we set the number of topics N as 50, and define the context view window size as {-3,3} (i.e. previous 3 words and next 3 words) in the LaSA model learning. LaSA features for other irrespective words (e.g. token unit "the") are assigned with a default topic value N+1.

All the basic NER models are trained on the domain-specific training data using RRM classifier (Guo et al., 2005). RRM is a generalization Winnow learning algorithm (Zhang et al., 2002). We set the context view window size as {-2,2} in NER. Given a word instance x, we employ local linguistic features (e.g. word unit, part of speech) of x and its context units (i.e. previous 2 words and next 2 words) in NER. All Chinese texts in the experiments are automatically segmented into words using HMM.

In LaSA-based domain adaptation, the semantic association features of each unit in the observation window {-2,2} are generated by LaSA model at first, then the basic source domain NER model is tuned on the original source domain training data set by incorporating the semantic association features. For example, given the sentence *"This popular new singer attended the new year party"*, Figure 1 illustrates various features and views at the current word w_i= "$singer$" in LaSA-based adaptation.

			→ Tagging →		
Position	w_{i-2}	w_{i-1}	w_i	w_{i+1}	w_{i+2}
Word	$popular$	new	$singer$	$attend$	the
POS	adj	adj	$noun$	$verb$	$article$
SA	$SA(popular)$	$SA(new)$	$SA(singer)$	$SA(attend)$	$SA(the)$
.....					
Tag	t_{i-2}	t_{i-1}	$\mathbf{t_i}$		

Figure 1: Feature window in LaSA-based adaptation

In the viewing window at the word "$singer$" (see Figure 1), each word unit around "$singer$" is codified with a set of primitive features (e.g. POS, SA, Tag), together with its relative position to "$singer$".

Here, "SA" denotes semantic association feature set which is generated by LaSA model. "Tag" denotes NE tags labeled in the data set.

Given the input vector constructed with the above features, RRM method is then applied to train linear weight vectors, one for each possible class-label. In the decoding stage, the class with the maximum confidence is then selected for each token unit.

In our evaluation, only NEs with correct boundaries and correct class labels are considered as the correct recognition. We use the standard Precision (P), Recall (R), and F-measure ($F = \frac{2PR}{P+R}$) to measure the performance of NER models.

5.2 Data

We built large-scale English and Chinese annotated corpus. English corpus are generated from wikipedia while Chinese corpus are selected from Chinese newspapers. Moreover, test data do not overlap with training data and unlabeled data.

5.2.1 Generate English Annotated Corpus from Wikipedia

Wikipedia provides a variety of data resources for NER and other NLP research (Richman and Schone, 2008). We generate all the annotated English corpus from wikipedia. With the limitation of efforts, only PER NEs in the corpus are automatically tagged using an English person gazetteer. We automatically extract an English Person gazetteer from wikipedia at first. Then we select the articles from wikipedia and tag them using this gazetteer.

In order to build the English Person gazetteer from wikipdedia, we manually selected several key phrases, including "births", "deaths", "surname", "given names" and "human names" at first. For each article title of interest, we extracted the categories to which that entry was assigned. The entry is considered as a person name if its related explicit category links contain any one of the key phrases, such as "Category: human names". We totally extracted 25,219 person name candidates from 204,882 wikipedia articles. And we expanded this gazetteer by adding the other available common person names. Finally, we obtained a large-scale gazetteer of 51,253 person names.

All the articles selected from wikipedia are further tagged using the above large-scale gazetteer. Since human annotated set were not available, we held out more than 100,000 words of text from the automatically tagged corpus to as a test set in each domain. Table 2 shows the data distribution of the training and test data sets.

Domains	Training Data Set		Test Data Set	
	Size	PERs	Size	PERs
Pol	0.45M	9,383	0.23M	6,067
Eco	1.06M	21,023	0.34M	6,951
Spo	0.47M	17,727	0.20M	6,075
Ent	0.36M	12,821	0.15M	5,395

Table 2: English training and test data sets

We also randomly select 17M unlabeled English data (see Table 3) from Wikipedia. These unlabeled data are used to build the English LaSA model.

	All	Domain			
		Pol	Eco	Spo	Ent
Data Size(M)	17.06	7.36	2.59	3.65	3.46

Table 3: Domain distribution in the unlabeled English data set

5.2.2 Chinese Data

We built a large-scale high-quality Chinese NE annotated corpus. All the data are news articles from several Chinese newspapers in 2001 and 2002. All the NEs (i.e. PER, LOC and ORG) in the corpus are manually tagged. Cross-validation checking is employed to ensure the quality of the annotated corpus.

Domain	Size	NEs in the training data set			
	(M)	PER	ORG	LOC	Total
Pol	0.90	11,388	6,618	14,350	32,356
Eco	1.40	6,821	18,827	14,332	39,980
Spo	0.60	11,647	8,105	7,468	27,220
Ent	0.60	12,954	2,823	4,665	20,442
Domain	Size	NEs in the test data set			
	(M)	PER	ORG	LOC	Total
Pol	0.20	2,470	1,528	2,540	6,538
Eco	0.26	1,098	2,971	2,362	6,431
Spo	0.10	1,802	1,323	1,246	4,371
Ent	0.10	2,458	526	738	3,722

Table 4: Chinese training and test data sets

All the domain-specific training and test data are selected from this annotated corpus according to the domain categories (see Table 4). 8.46M unlabeled Chinese data (see Table 5) are randomly selected from this corpus to build the Chinese LaSA model.

5.3 Experimental Results

All the experiments are conducted on the above large-scale English and Chinese corpus. The overall performance enhancement of NER by LaSA-based

	All	Domain			
		Pol	Eco	Spo	Ent
Data Size(M)	8.46	2.34	1.99	2.08	2.05

Table 5: Domain distribution in the unlabeled Chinese data set

domain adaptation is evaluated at first. Since the distribution of each NE type is different across domains, we also analyze the performance enhancement on each entity type by LaSA-based adaptation.

5.3.1 Performance Enhancement of NER by LaSA-based Domain Adaptation

Table 6 and 7 show the experimental results for all pairs of domain adaptation on both English and Chinese corpus, respectively. In the experiment, the basic source domain NER model M_s is learned from the specific domain training data set D_{dom} (see Table 2 and 4 in Section 5.2). Here, $dom \in \{Eco, Ent, Pol, Spo\}$. F^{in}_{dom} denotes the top-line F-measure of M_s in the source trained domain dom. When M_s is directly applied in a new target domain, its F-measure in this basic transfer is considered as baseline (denoted by F_{Base}). F_{LaSA} denotes F-measure of M_s achieved in the target domain with LaSA-based domain adaptation. $\delta(F) = \frac{F_{LaSA} - F_{Base}}{F_{Base}}$, which denotes the relative F-measure enhancement by LaSA-based domain adaptation.

Source → Target	Performance in the domain transfer				
	F_{Base}	F_{LaSA}	$\delta(F)$	$\delta(loss)$	F_{Top}
Eco→Ent	57.61%	59.22%	+2.79%	17.87%	F^{in}_{Ent}=66.62%
Pol→Ent	57.5 %	59.83%	+4.05%	25.55%	F^{in}_{Ent}=66.62%
Spo→Ent	58.66%	62.46%	+6.48%	47.74%	F^{in}_{Ent}=66.62%
Ent→Eco	70.56 %	72.46%	+2.69%	19.33%	F^{in}_{Eco}=80.39%
Pol→Eco	63.62%	68.1%	+7.04%	26.71%	F^{in}_{Eco}=80.39%
Spo→Eco	70.35%	72.85%	+3.55%	24.90%	F^{in}_{Eco}=80.39%
Eco→Pol	50.59%	52.7%	+4.17%	15.81%	F^{in}_{Pol}=63.94%
Ent→Pol	56.12%	59.82%	+6.59%	47.31%	F^{in}_{Pol}=63.94%
Spo→Pol	60.22%	62.6%	+3.95%	63.98%	F^{in}_{Pol}=63.94%
Eco→Spo	60.28%	61.21%	+1.54%	9.93%	F^{in}_{Spo}=69.65%
Ent→Spo	60.28%	62.68%	+3.98%	25.61%	F^{in}_{Spo}=69.65%
Pol→Spo	56.94%	60.48%	+6.22%	27.85%	F^{in}_{Spo}=69.65%

Table 6: Experimental results on English corpus

Experimental results on English and Chinese corpus indicate that the performance of M_s significantly degrades in each basic domain transfer without using LaSA model (see Table 6 and 7). For example, in the "Eco→Ent" transfer on Chinese corpus (see Table 7), F^{in}_{eco} of M_s is 82.28% while F_{Base} of M_s is 60.45% in the entertainment domain. F-measure of M_s significantly degrades by 21.83 percent points in this basic transfer. Significant performance degrading of M_s is observed in all the basic transfer. It shows that the data distribution of both domains is very different in each possible transfer.

Source → Target	Performance in the domain transfer				
	F_{Base}	F_{LaSA}	$\delta(F)$	$\delta(loss)$	F_{Top}
Eco→Ent	60.45%	66.42%	+9.88%	26.29%	F^{in}_{Ent}=83.16%
Pol→Ent	69.89%	73.07%	+4.55%	23.96%	F^{in}_{Ent}=83.16%
Spo→Ent	68.66%	70.89%	+3.25%	15.38%	F^{in}_{Ent}=83.16%
Ent→Eco	58.50%	61.35%	+ 4.87%	11.98%	F^{in}_{Eco}=82.28%
Pol→Eco	62.89%	64.93%	+3.24%	10.52%	F^{in}_{Eco}=82.28%
Spo→Eco	60.44%	63.20%	+ 4.57 %	12.64%	F^{in}_{Eco}=82.28%
Eco→Pol	67.03%	70.90 %	+5.77%	27.78%	F^{in}_{Pol}=80.96%
Ent→Pol	66.64 %	68.94 %	+3.45%	16.06%	F^{in}_{Pol}=80.96%
Spo→Pol	65.40%	67.20%	+2.75%	11.57%	F^{in}_{Pol}=80.96%
Eco→Spo	67.20%	70.77%	+5.31%	15.47%	F^{in}_{Spo}=90.24%
Ent→Spo	70.05%	72.20%	+3.07%	10.64%	F^{in}_{Spo}=90.24%
Pol→Spo	70.99%	73.86%	+4.04%	14.91%	F^{in}_{Spo}=90.24%

Table 7: Experimental results on Chinese corpus

Experimental results on English corpus show that LaSA-based adaptation effectively enhances the performance in each domain transfer (see Table 6). For example, in the "Pol→Eco" transfer, F_{Base} is 63.62% while F_{LaSA} achieves 68.10%. Compared with F_{Base}, LaSA-based method significantly enhances F-measure by 7.04%. We perform t-tests on F-measure of all the comparison experiments on English corpus. The p-value is 2.44E-06, which shows that the improvement is statistically significant.

Table 6 also gives the accuracy loss due to transfer in each domain adaptation on English corpus. The accuracy loss is defined as $loss = 1 - \frac{F}{F^{in}_{dom}}$. And the relative reduction in error is defined as $\delta(loss)$= $|1 - \frac{loss_{LaSA}}{loss_{Base}}|$. Experimental results indicate that the relative reduction in error is above 9.93% with LaSA-based transfer in each test on English corpus. LaSA model significantly decreases the accuracy loss by 29.38% in average. Especially for "Spo→Pol" transfer, $\delta(loss)$ achieves 63.98% with LaSA-based adaptation. All the above results show that LaSA-based adaptation significantly reduces the accuracy loss in the domain transfer for English NER without any labeled target domain samples.

Experimental results on Chinese corpus also show that LaSA-based adaptation effectively increases the accuracy in all the tests (see Table 7). For example, in the "Eco→Ent" transfer, compared with F_{Base}, LaSA-based adaptation significantly increases F-measure by 9.88%. We also perform t-tests on F-

measure of 12 comparison experiments on Chinese corpus. The p-value is 1.99E-06, which shows that the enhancement is statistically significant. Moreover, the relative reduction in error is above 10% with LaSA-based method in each test. LaSA model decreases the accuracy loss by 16.43% in average. Especially for the "Eco→Ent" transfer (see Table 7), $\delta(loss)$ achieves 26.29% with LaSA-based method.

All the above experimental results on English and Chinese corpus show that LaSA-based domain adaptation significantly decreases the accuracy loss in the transfer without any labeled target domain data. Although automatically tagging introduced some errors in English source training data, the relative reduction in errors in English NER adaptation seems comparable to that one in Chinese NER adaptation.

5.3.2 Accuracy Enhancement for Each NE Type Recognition

Our statistic data (Guo et al., 2006) show that the distribution of NE types varies with domains. Each NE type has different domain features. Thus, the performance stability of each NE type recognition is very important in the domain transfer.

Figure 2 gives F-measure of each NE type recognition achieved by LaSA-based adaptation on English and Chinese corpus. Experimental results show that LaSA-based adaptation effectively increases the accuracy of each NE type recognition in the most of the domain transfer tests. We perform t-tests on F-measure of the comparison experiments on each NE type, respectively. All the p-value is less than 0.01, which shows that the improvement on each NE type recognition is statistically significant. Especially, the p-value of English and Chinese PER is 2.44E-06 and 9.43E-05, respectively, which shows that the improvement on PER recognition is very significant. For example, in the "Eco→Pol" transfer on Chinese corpus, compared with F_{Base}, LaSA-based adaptation enhances F-measure of PER recognition by 9.53 percent points. Performance enhancement for ORG recognition is less than that one for PER and LOC recognition using LaSA model since ORG NEs usually contain much more domain-specific information than PER and LOC.

The major reason for error reduction is that external context and internal units are better semantically associated using LaSA model. For example, LaSA

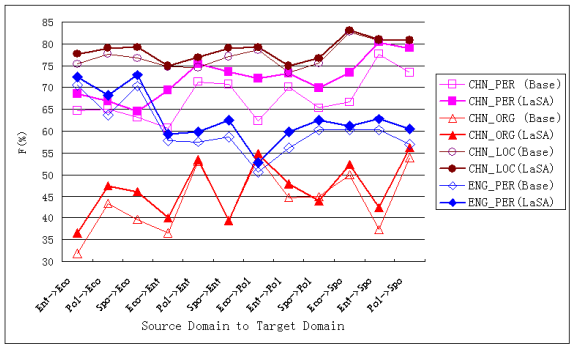


Figure 2: PER, LOC and ORG recognition in the transfer

model better groups various titles from different domains (see Table 1 in Section 4.2). Various industry terms in ORG NEs are also grouped into the semantic sets. These semantic associations provide useful hints for detecting the boundary of NEs in the new target domain. All the above results show that LaSA model better compensates for the feature distribution difference of each NE type across domains.

6 Conclusion

We present a domain adaptation method with LaSA model in this paper. LaSA model captures latent semantic association among words from the unlabeled corpus. It better groups words into a set of concepts according to the related context snippets. LaSA-based domain adaptation method projects words to a low-dimension concept feature space in the transfer. It effectively overcomes the data distribution gap across domains without using any labeled target domain data. Experimental results on English and Chinese corpus show that LaSA-based domain adaptation significantly enhances the performance of NER across domains. Especially, LaSA model effectively increases the accuracy of each NE type recognition in the domain transfer. Moreover, LaSA-based domain adaptation method works well across languages. To further reduce the accuracy loss, we will explore informative sampling to capture fine-grained data difference in the domain transfer.

References

Rie Ando and Tong Zhang. 2005. A Framework for Learning Predictive Structures from Multiple Tasks

and Unlabeled Data. In *Journal of Machine Learning Research 6 (2005)*, pages 1817–1853.

Andrew Arnold, Ramesh Nallapati, and William W. Cohen. 2008. Exploiting Feature Hierarchy for Transfer Learning in Named Entity Recognition. In *Proceedings of 46th Annual Meeting of the Association of Computational Linguistics (ACL'08)*, pages 245-253.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research, 3:993–1022*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain Adaptation with Structural Correspondence Learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 120-128.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*, pages 440-447.

Andrew Borthwick. 1999. *A Maximum Entropy Approach to Named Entity Recognition*. Ph.D. thesis, New York University.

Yee Seng Chan and Hwee Tou Ng. 2007. Domain Adaptation with Active Learning for Word Sense Disambiguation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*.

Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy capitalizer: Little data can help a lot. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information and lexicography. *Computational Linguistics*, 16(1):22–29.

Hal Daume III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Scott Deerwester, Susan T. Dumais, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recogintion through classifier combination. In *Proceedings of the 2003 Conference on Computational Natural Language Learning*.

Freitag. 2004. Trained Named Entity Recognition Using Distributional Clusters. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name Tagging with Word Clusters and Discriminative Training. In *Proceedings of HLT-NAACL 04*.

Jianfeng Gao, Mu Li, Anndy Wu, and Changning Huang. 2005. Chinese Word Segmentation and Named Entity Recognition: A Pragmatic Approach. *Computational Linguisitc*, 31(4):531–574.

Honglei Guo, Jianmin Jiang, Gang Hu, and Tong Zhang. 2005. Chinese Named Entity Recognition Based on Multilevel Linguistic Features. In *Lecture Notes in Artificial Intelligence*, 3248:90–99.

Honglei Guo, Li Zhang, and Zhong Su. 2006. Empirical Study on the Performance Stability of Named Entity Recognition Model across Domains. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 509-516.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22th Annual International SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*.

Jing Jiang and ChengXiang Zhai. 2006. Exploiting Domain Structure for Named Entity Recognition. In *Proceedings of HLT-NAACL 2006*, pages 74–81.

Jing Jiang and ChengXiang Zhai. 2007. Instance Weighting for Domain Adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL'07)*, pages 264–271.

Wei Li and Andrew McCallum. 2005. Semi-supervised sequence modeling with syntactic topic models. In *Proceedings of Twenty AAAI Conference on Artificial Intelligence (AAAI-05)*.

Alexander E. Richman and Patrick Schone. 2008. Mining Wiki Resources for Multilingual Named Entity Recognition. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*.

Brian Roark and Michiel Bacchiani. 2003. Supervised and unsupervised PCFG adaptation to novel domains. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language independent named entity recognition. In *Proceedings of the 2003 Conference on Computational Natural Language Learning (CoNLL-2003)*, pages 142–147.

Xing Wei and Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International SIGIR Conference on Research and Development in Information Retrieval*.

Tong Zhang, Fred Damerau, and David Johnson. 2002 Text chunking based on a generalization of Winnow. *Journal of Machine Learning Research*, 2:615–637.

Semi-Automatic Entity Set Refinement

Vishnu Vyas and Patrick Pantel
Yahoo! Labs
Santa Clara, CA 95054
{vishnu,ppantel}@yahoo-inc.com

Abstract

State of the art set expansion algorithms produce varying quality expansions for different entity types. Even for the highest quality expansions, errors still occur and manual refinements are necessary for most practical uses. In this paper, we propose algorithms to aide this refinement process, greatly reducing the amount of manual labor required. The methods rely on the fact that most expansion errors are systematic, often stemming from the fact that some seed elements are ambiguous. Using our methods, empirical evidence shows that average R-precision over random entity sets improves by 26% to 51% when given from 5 to 10 manually tagged errors. Both proposed refinement models have linear time complexity in set size allowing for practical online use in set expansion systems.

1 Introduction

Sets of named entities are extremely useful in a variety of natural language and information retrieval tasks. For example, companies such as Yahoo! and Google maintain sets of named entities such as cities, products and celebrities to improve search engine relevance.

Manually creating and maintaining large sets of named entities is expensive and laborious. In response, many automatic and semi-automatic methods of creating sets of named entities have been proposed, some are supervised (Zhou and Su, 2001), unsupervised (Pantel and Lin 2002, Nadeau et al. 2006), and others semi-supervised (Kozareva et al. 2008). Semi-supervised approaches are often used in practice since they allow for targeting specific entity classes such as *European Cities* and *French Impressionist Painters*. Methods differ in complexity from simple ones using lexico-syntactic patterns (Hearst 1992) to more complicated techniques based on distributional similarity (Paşca 2007a).

Even for state of the art methods, expansion errors inevitably occur and manual refinements are necessary for most practical uses requiring high precision (such as for query interpretation at commercial search engines). Looking at expansions from state of the art systems such as GoogleSets[1], we found systematic errors such as those resulting from ambiguous seed instances. For example, consider the following seed instances for the target set *Roman Gods*:

`Minerva, Neptune, Baccus, Juno, Apollo`

GoogleSet's expansion as well others employing distributional expansion techniques consists of a mishmash of Roman Gods and celestial bodies, originating most likely from the fact that *Neptune* is both a *Roman God* and a *Planet*. Below is an excerpt of the GoogleSet expansion:

`Mars, Venus, *Moon, Mercury, *asteroid, Jupiter, *Earth, *comet, *Sonne, *Sun, …`

The inherent semantic similarity between the errors can be leveraged to quickly clean up the expansion. For example, given a manually tagged error "*asteroid*", a distributional similarity thesaurus

[1] http://labs.google.com/sets

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 290–298,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

such as (Lin 1998)[2] can identify *comet* as similar to *asteroid* and therefore potentially also as an error. This method has its limitations since a manually tagged error such as *Earth* would correctly remove *Moon* and *Sun*, but it would also incorrectly remove *Mars*, *Venus* and *Jupiter* since they are also similar to *Earth*[3].

In this paper, we propose two algorithms to improve the precision of automatically expanded entity sets by using minimal human negative judgments. The algorithms leverage the fact that set expansion errors are systematically caused by ambiguous seed instances which attract incorrect instances of an unintended entity type. We use distributional similarity and sense feature modeling to identify such unintended entity types in order to quickly clean up errors with minimal manual labor. We show empirical evidence that average R-precision over random entity sets improves by 26% to 51% when given from 5 to 10 manually tagged errors. Both proposed refinement models have linear time complexity in set size allowing for practical online use in set expansion systems.

The remainder of this paper is organized as follows. In the next section we review related work and position our contribution within its landscape. Section 3 presents our task of dynamically modeling the similarity of a set of words and describes algorithms for refining sets of named entities. The datasets and our evaluation methodology used to perform our experiments are presented in Section 4 and in Section 5 we describe experimental results. Finally, we conclude with some discussion and future work.

2 Related Work

There is a large body of work for automatically building sets of named entities using various techniques including supervised, unsupervised and semi-supervised methods. Supervised techniques use large amounts of training data to detect and classify entities into coarse grained classes such as *People*, *Organizations*, and *Places* (Bunescu and Mooney 2004; Etzioni et al. 2005). On the other hand, unsupervised methods require no training data and rely on approaches such as clustering, targeted patterns and co-occurrences to extract sets of entities (Pantel and Lin 2002; Downey et al. 2007).

Semi-supervised approaches are often used in practice since they allow for targeting specific entity classes. These methods rely on a small set of seed examples to extract sets of entities. They either are based on distributional approaches or employ lexico-syntactic patterns to expand a small set of seeds to a larger set of candidate expansions. Some methods such as (Riloff and Shepherd 1997; Riloff and Jones 1999; Banko et al. 2007;Paşca 2007a) use lexico-syntactic patterns to expand a set of seeds from web text and query logs. Others such as (Paşca et al. 2006; Paşca 2007b; Paşca and Durme 2008) use distributional approaches. Wang and Cohen (2007) use structural cues in semi-structured text to expand sets of seed elements. In all methods however, expansion errors inevitably occur. This paper focuses on the task of post processing any such system's expansion output using minimal human judgments in order to remove expansion errors.

Using user feedback to improve a system's performance is a common theme within many information retrieval and machine learning tasks. One form of user feedback is active learning (Cohn et al. 1994), where one or more classifiers are used to focus human annotation efforts on the most beneficial test cases. Active learning has been successfully applied to various natural language tasks such as parsing (Tang et al. 2001), POS tagging (Dagan and Engelson 1995) and providing large amounts of annotations for common natural language processing tasks such as word sense disambiguation (Banko and Brill 2001). Relevance feedback is another popular feedback paradigm commonly used in information retrieval (Harman 1992), where user feedback (either explicit or implicit) is used to refine the search results of an IR system. Relevance feedback has been successfully applied to many IR applications including content-based image retrieval (Zhouand Huang 2003) and web search (Vishwa et al. 2005). Within NLP applications relevance feedback has also been used to generate sense tagged examples for WSD tasks (Stevenson et al. 2008), and Question Answering (Negri 2004). Our methods use relevance feedback in the form of negative examples to refine the results of a set expansion system.

[2] See http://demo.patrickpantel.com/ for a demonstration of the distributional thesaurus.

[3] In practice, this problem is rare since most terms that are similar in one of their senses tend not to be similar in their other senses.

3 Dynamic Similarity Modeling

The set expansion algorithms discussed in Section 2 often produce high quality entity sets, however inevitably errors are introduced. Applications requiring high precision sets must invest significantly in editorial efforts to clean up the sets. Although companies like Yahoo! and Google can afford to routinely support such manual labor, there is a large opportunity to reduce the refinement cost (i.e., number of required human judgments).

Recall the set expansion example of *Roman Gods* from Section 1. Key to our approach is the hypothesis that *most expansion errors result from some systematic cause*. Manual inspection of expansions from GoogleSets and distributional set expansion techniques revealed that most errors are due to the inherent ambiguity of seed terms (such as *Neptune* in our example) and data sparseness (such as *Sonne* in our example, a very rare term). The former kind of error is systematic and can be leveraged by an automatic method by assuming that any entity semantically similar to an identified error will also be erroneous.

In this section, we propose two methods for leveraging this hypothesis. In the first method, described in Section 3.1, we use a simple distributional thesaurus and remove all entities which are distributionally similar to manually identified errors. In the second method, described in Section 3.2, we model the semantics of the seeds using distributional features and then dynamically change the feature space according to the manually identified errors and rerank the entities in the set. Both methods rely on the following two observations:

a) **Many expansion errors are systematically caused by ambiguous seed examples** which draw in several incorrect entities of its unintended senses (such as seed *Neptune* in our *Roman Gods* example which drew in celestial bodies such as *Earth* and *Sun*);
b) **Entities which are similar in one sense are usually not similar in their other senses**. For example, *Apple* and *Sun* are similar in their *Company* sense but their other senses (*Fruit* and *Celestial Body*) are not similar. Our example in Section 1 illustrates a rare counterexample where *Neptune* and *Mercury* are similar in both their *Planets* and *Roman Gods* senses.

Task Outline: Our task is to remove errors from entity sets by using a minimal amount of manual judgments. Incorporating feedback into this process can be done in multiple ways. The most flexible system would allow a judge to iteratively remove as many errors as desired and then have the system automatically remove other errors in each iteration. Because it is intractable to test arbitrary numbers of manually identified errors in each iteration, we constrain the judge to identify at most one error in each iteration.

Although this paper focuses solely on removing errors in an entity set, it is also possible to improve expanded sets by using feedback to add new elements to the sets. We consider this task out of scope for this paper.

3.1 Similarity Method (SIM)

Our first method directly models observation *a*) in the previous section. Following Lin (1998), we model the similarity between entities using the distributional hypothesis, which states that similar terms tend to occur in similar contexts (Harris 1985). A semantic model can be obtained by recording the surrounding contexts for each term in a large collection of unstructured text. Methods differ in their definition of a context (e.g., *text window* or *syntactic relations*), or a means to weigh contexts (e.g., *frequency*, *tf-idf*, *pointwise mutual information*), or ultimately in measuring the similarity between two context vectors (e.g., using *Euclidean distance*, *Cosine*, *Dice*). In this paper, we use a text window of size 1, we weigh our contexts using pointwise mutual information, and we use the cosine score to compute the similarity between context vectors (i.e., terms). Section 5.1 describes our source corpus and extraction details. Computing the full similarity matrix for many terms over a very large corpus is computationally intensive. Our specific implementation follows the one presented in (Bayardo et al. 2007).

The similarity matrix computed above is then directly used to refine entity sets. Given a manually identified error at each iteration, we automatically remove each entity in the set that is found to be semantically similar to the error. The similarity threshold was determined by manual inspection and is reported in Section 5.1.

Due to observation *b*) in the previous section, we expect that this method will perform poorly on

entity sets such as the one presented in our example of Section 1 where the manual removal of *Earth* would likely remove correct entities such as *Mars*, *Venus* and *Jupiter*. The method presented in the next section attempts to alleviate this problem.

3.2 Feature Modification Method (FMM)

Under the distributional hypothesis, the semantics of a term are captured by the contexts in which it occurs. The *Feature Modification Method* (FMM), in short, attempts to automatically discover the incorrect contexts of the unintended senses of seed elements and then filters out expanded terms whose contexts do not overlap with the other contexts of the seed elements.

Consider the set of seed terms *S* and an erroneous expanded instance *e*. In the SIM method of Section 3.1 all set elements that have a feature vector (i.e., context vector) similar to *e* are removed. The *Feature Modification Method* (FMM) instead tries to identify the subset of features of the error *e* which represent the unintended sense of the seed terms *S*. For example, let *S* = {*Minerva*, *Neptune*, *Baccus*, *Juno*, *Apollo*}. Looking at the contexts of these words in a large corpus, we construct a centroid context vector for *S* by taking a weighted average of the contexts of the seeds in *S*. In Wikipedia articles we see contexts (i.e., features) such as[4]:

```
attack, kill, *planet, destroy,
Goddess, *observe, statue, *launch,
Rome, *orbit, …
```

Given an erroneous expansion such as *e* = *Earth*, we postulate that removing the intersecting features from *Earth*'s feature vector and the above feature vector will remove the unintended *Planet* sense of the seed set caused by the seed element *Neptune*. The intersecting features that are removed are bolded in the above feature vector for *S*. The similarity between this modified feature vector for *S* and all entities in the expansion set can be recomputed as described in Section 3.1. Entities with a low similarity score are removed from the expanded set since they are assumed to be part of the unintended semantic class (*Planet* in this example).

Unlike the SIM method from Section 3.1, this method is more stable with respect to observation *b*) in Section 3. We showed that SIM would incorrectly remove expansions such as *Mars*, *Venus* and *Jupiter* given the erroneous expansion *Earth*. The FMM method would instead remove the *Planet* features from the seed feature vectors and the remaining features would still overlap with *Mars*, *Venus* and *Jupiter*'s *Roman God* sense.

Efficiency: FMM requires online similarity computations between centroid vectors and all elements of the expanded set. For large corpora such as Wikipedia articles or the Web, feature vectors are large and storing them in memory and performing similarity computations repeatedly for each editorial judgment is computationally intensive. For example, the size of the feature vector for a single word extracted from Wikipedia can be in the order of a few gigabytes. Storing the feature vectors for all candidate expansions and the seed set is inefficient and too slow for an interactive system. The next section proposes a solution that makes this computation very fast, requires little memory, and produces near perfect approximations of the similarity scores.

3.3 Approximating Cosine Similarity

There are engineering optimizations that are available that allow us to perform a near perfect approximation of the similarity computation from the previous section. The proposed method requires us to only store the shared features between the centroid and the words rather than the complete feature vectors, thus reducing our space requirements dramatically. Also, FMM requires us to repeatedly calculate the cosine similarity between a modified centroid feature vector and each candidate expansion at each iteration. Without the full context vectors of all candidate expansions, computing the exact cosine similarity is impossible. Given, however, the original cosine scores between the seed elements and the candidate expansions before the first refinement iteration as well as the shared features, we can approximate with very high accuracy the updated cosine score between the modified centroid and each candidate expansion. Our method relies on the fact that features (i.e., contexts) are only ever removed from the original centroid – no new features are ever added.

Let μ be the original centroid representing the seed instances. Given an expansion error *e*, FMM creates a modified centroid by removing all fea-

[4] The full feature vector for these and all other terms in Wikipedia can be found at http://demo.patrickpantel.com/..

tures intersecting between e and μ. Let μ' be this modified centroid. FMM requires us to compute the similarity between μ' and all candidate expansions x as:

$$\cos(x,\mu') = \frac{\sum x_i \mu'_i}{\|x\| \cdot \|\mu'\|}$$

where i iterates over the feature space.

In our efficient setting, the only element that we do not have for calculating the exact cosine similarity is the norm of x, $\|x\|$. Given that we have the original cosine similarity score, $cos(x, \mu)$ and that we have the shared features between the original centroid μ and the candidate expansion x we can calculate $\|x\|$ as:

$$\|x\| = \frac{\sum x_i \mu_i}{\|\mu\| \cdot \cos(x,\mu)}$$

Combining the two equations, have:

$$\cos(x,\mu') = \cos(x,\mu) \cdot \frac{\sum x_i \mu'_i}{\sum x_i \mu_i} \cdot \frac{\|\mu\|}{\|\mu'\|}$$

In the above equation, the modified cosine score can be considered as an update to the original cosine score, where the update depends only on the shared features and the original centroid. The above update equation can be used to recalculate the similarity scores without resorting to an expensive computation involving complete feature vectors.

Storing the original centroid is expensive and can be approximated instead from only the shared features between the centroid and all instances in the expanded set. We empirically tested this approximation by comparing the cosine scores between the candidate expansions and both the true centroid and the approximated centroid. The average error in cosine score was 9.5E-04 ± 7.83E-05 (95% confidence interval).

4 Datasets and Baseline Algorithm

We evaluate our algorithms against manually scraped gold standard sets, which were extracted from Wikipedia to represent a random collection of concepts. Section 4.1 discusses the gold standard sets and the criteria behind their selection. To present a statistically significant view of our results we generated a set of trials from gold standard sets to use as seeds for our seed set expansion algorithm. Also, in section 4.2 we discuss how we can simulate editorial feedback using our gold standard sets.

4.1 Gold Standard Entity Sets

The gold standard sets form an essential part of our evaluation. These sets were chosen to represent a single concept such as *Countries* and *Archbishops of Canterbury*. These sets were selected from the *List of* pages from Wikipedia[5]. We randomly sorted the list of every noun occurring in Wikipedia. Then, for each noun we verified whether or not it existed in a Wikipedia list, and if so we extracted this list – up to a maximum of 50 lists. If a noun belonged to multiple lists, the authors chose the list that seemed most appropriate. Although this does not generate a perfect random sample, diversity is ensured by the random selection of nouns and relevancy is ensured by the author adjudication.

Lists were then scraped from the Wikipedia website and they went through a manual cleanup process which included merging variants. . The 50 sets contain on average 208 elements (with a minimum of 11 and a maximum of 1116 elements) for a total of 10,377 elements. The final gold standard lists contain 50 sets including *classical pianists*, *Spanish provinces*, *Texas counties*, *male tennis players*, *first ladies*, *cocktails*, *bottled water brands*, and *Archbishops of Canterbury*[6].

4.2 Generation of Experimental Trials

To provide a statistically significant view of the performance of our algorithm, we created more than 1000 trials as follows. For each of the gold standard seed sets, we created 30 random sortings. These 30 random sortings were then used to generate trial seed sets with a maximum size of 20 seeds.

4.3 Simulating User Feedback and Baseline Algorithm

User feedback forms an integral part of our algorithm. We used the gold standard sets to judge the

[5] In this paper, extractions from Wikipedia are taken from a snapshot of the resource in December 2007.

[6] The gold standard is available for download at http://www.patrickpantel.com/cgi-bin/Web/Tools/getfile.pl?type=data&id=sse-gold/wikipedia.20071218.goldsets.tgz

candidate expansions. The judged expansions were used to simulate user feedback by marking those candidate expansions that were incorrect. The first candidate expansion that was marked incorrect in each editorial iteration was used as the editor's negative example and was given to the system as an error.

In the next section, we report R-precision gains at each iteration in the editorial process for our two methods described in Section 3. Our baseline method simply measures the gains obtained by removing the first incorrect entry in a candidate expansion set at each iteration. This simulates the process of manually cleaning a set by removing one error at a time.

Table 1. R-precision of the three methods with 95% confidence bounds.

ITERATION	*BASELINE*	*SIM*	*FMM*
1	0.219±0.012	**0.234±0.013**	0.220±0.015
2	0.223±0.013	**0.242±0.014**	0.227±0.017
3	0.227±0.013	**0.251±0.015**	0.235±0.019
4	0.232±0.013	**0.26±0.016**	0.252±0.021
5	0.235±0.014	**0.266±0.017**	**0.267±0.022**
6	0.236±0.014	0.269±0.017	**0.282±0.023**
7	0.238±0.014	0.273±0.018	**0.294±0.023**
8	0.24±0.014	0.28±0.018	**0.303±0.024**
9	0.242±0.014	0.285±0.018	**0.315±0.025**
10	0.243±0.014	0.286±0.018	**0.322±0.025**

5 Experimental Results

5.1 Experimental Setup

Wikipedia[5] served as the source corpus for our algorithms described in Sections 3.1 and 3.2. All articles were POS-tagged using (Brill 1995) and later chunked using a variant of (Abney 1991). Corpus statistics from this processed text were collected to build the similarity matrix for the SIM method (Section 3.1) and to extract the features required for the FMM method (Section 3.2). In both cases corpus statistics were extracted over the semi-syntactic contexts (chunks) to approximate term meanings. The minimum similarity thresholds were experimentally set to 0.15 and 0.11 for the SIM and FMM algorithms respectively.

Each experimental trial described in Section 4.2, which consists of a set of seed instances of one of our 50 random semantic classes, was expanded using a variant of the distributional set expansion algorithm from Sarmento et al. (2007). The expansions were judged against the gold standard and each candidate expansion was marked as either correct or incorrect. This set of expanded and judged candidate files were used as inputs to the algorithms described in Sections 3.1 and 3.2. Choosing the first candidate expansion that was judged as incorrect simulated our user feedback. This process was repeated for each iteration of the algorithm and results are reported for 10 iterations.

The outputs of our algorithms were again judged against the gold standard lists and the performance was measured in terms of precision gains over the baseline at various ranks. Precision gain for an algorithm over a baseline is the percentage increase in precision for the same values of parameters of the algorithm over the baseline. Also, as the size of our gold standard lists vary, we report another commonly used statistic, R-precision. R-precision for any set is the precision at the size of the gold standard set. For example, if a gold standard set contains 20 elements, then R-precision for any set expansion is measured as the precision at rank 20. The average R-precision over each set is then reported.

5.2 Quantitative Analysis

Table 1 lists the performance of our baseline algorithm (Section 4.3) and our proposed methods SIM and FMM (Sections 3.1 and 3.2) in terms of their R-precision with 95% confidence bounds over 10 iterations of each algorithm.

The FMM of Section 3.2 is the best performing method in terms of R-precision reaching a maximum value of 0.322 after the 10^{th} iteration. For small numbers of iterations, however, the SIM method outperforms FMM since it is bolder in its refinements by removing all elements similar to the tagged error. Inspection of FMM results showed that bad instances get ranked lower in early iterations but it is only after 4 or 5 iterations that they get pushed passed the similarity threshold (accounting for the low marginal increase in precision gain for FMM in the first 4 to 5 iterations).

FMM outperforms the SIM method by an average of 4% increase in performance (13% improvement after 10 iterations). However both the FMM and the SIM method are able to outperform

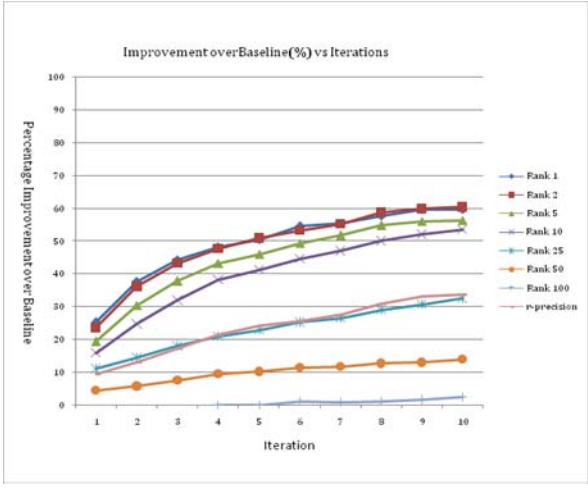


Figure 1. Precision gain over baseline algorithm for SIM method.

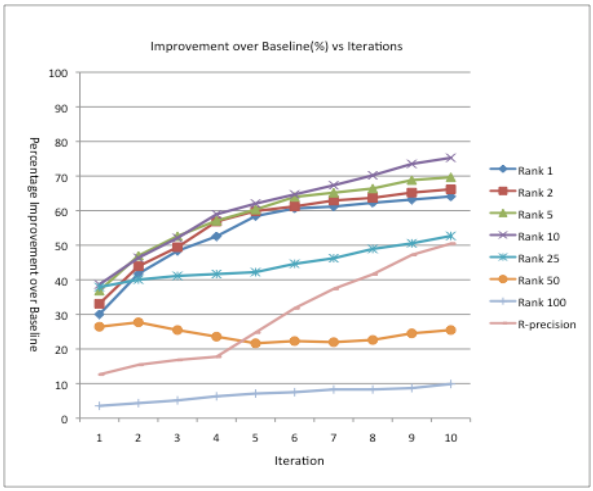


Figure 2. Precision gain over baseline algorithm for FMM method.

the baseline method. Using the FMM method one would achieve an average of 17% improvement in R-precision over manually cleaning up the set (32.5% improvement after 10 iterations). Using the SIM method one would achieve an average of 13% improvement in R-precision over manually cleaning up the set (17.7% improvement after 10 iterations).

5.3 Intrinsic Analysis of the SIM Algorithm

Figure 1 shows the precision gain of the similarity matrix based algorithm over the baseline algorithm. The results are shown for precision at ranks 1, 2, 5, 10, 25, 50 and 100, as well as for R-precision. The results are also shown for the first 10 iterations of the algorithm.

SIM outperforms the baseline algorithm for all ranks and increases in gain throughout the 10 iterations. As the number of iterations increases the change in precision gain levels off. This behavior can be attributed to the fact that we start removing errors from top to bottom and in each iteration the rank of the error candidate provided to the algorithm is lower than in the previous iteration. This results in errors which are not similar to any other candidate expansions. These are random errors and the discriminative capacity of this method reduces severely.

Figure 1 also shows that the precision gain of the similarity matrix algorithm over the baseline algorithm is higher at ranks 1, 2 and 5. Also, the performance increase drops at ranks 50 and 100. This is because low ranks contain candidate expansions that are random errors introduced due to data sparsity. Such unsystematic errors are not detectable by the SIM method.

5.4 Intrinsic Analysis of the FMM Algorithm

The feature modification method of Section 3.2 shows similar behavior to the SIM method, however as Figure 2 shows, it outperforms SIM method in terms of precision gain for all values of ranks tested. This is because the FMM method is able to achieve fine-grained control over what it removes and what it doesn't, as described in Section 5.2.

Another interesting aspect of FMM is illustrated in the R-precision curve. There is a sudden jump in precision gain after the fifth iteration of the algorithm. In the first iterations only few errors are pushed beneath the similarity threshold as centroid features intersecting with tagged errors are slowly removed. As the feature vector for the centroid gets smaller and smaller, remaining features look more and more unambiguous to the target entity type and erroneous example have less chance of overlapping with the centroid causing them to be pushed pass the conservative similarity threshold. Different conservative thresholds yielded similar curves. High thresholds yield bad performance since all but the only very prototypical set instances are removed as errors.

The R-precision measure indirectly models recall as a function of the target coverage of each set. We also directly measured recall at various ranks

and FMM outperformed SIM at all ranks and iterations.

5.5 Discussion

In this paper we proposed two techniques which use user feedback to remove systematic errors in set expansion systems caused by ambiguous seed instances. Inspection of expansion errors yielded other types of errors.

First, model errors are introduced in candidate expansion sets by noise from various preprocessing steps involved in generating the expansions. Such errors cause incorrect contexts (or features) to be extracted for seed instances and ultimately can cause erroneous expansions to be produced. These errors do not seem to be systematic and are hence not discoverable by our proposed method.

Other errors are due to data sparsity. As the feature space can be very large, the difference in similarity between a correct candidate expansion and an incorrect expansion can be very small for sparse entities. Previous approaches have suggested removing candidate expansions for which too few statistics can be extracted, however at the great cost of recall (and lower R-precision).

6 Conclusion

In this paper we presented two algorithms for improving the precision of automatically expanded entity sets by using minimal human negative judgments. We showed that systematic errors which arise from the semantic ambiguity inherent in seed instances can be leveraged to automatically refine entity sets. We proposed two techniques: SIM which boldly removes instances that are distributionally similar to errors, and FMM which more conservatively removes features from the seed set representing its unintended (ambiguous) concept in order to rank lower potential errors.

We showed empirical evidence that average R-precision over random entity sets improves by 26% to 51% when given from 5 to 10 manually tagged errors. These results were reported by testing the refinement algorithms on a set of 50 randomly chosen entity sets expanded using a state of the art expansion algorithm. Given very small amounts of manual judgments, the SIM method outperformed FMM (up to 4 manual judgments). FMM outperformed the SIM method given more than 6 manual judgments. Both proposed refinement models have linear time complexity in set size allowing for practical online use in set expansion systems.

This paper only addresses techniques for removing erroneous entities from expanded entity sets. A complimentary way to improve performance would be to investigate the addition of relevant candidate expansions that are not already in the initial expansion. We are currently investigating extensions to FMM that can efficiently add new candidate expansions to the set by computing the similarity between modified centroids and all terms occurring in a large body of text.

We are also investigating ways to use the findings of this work to a priori remove ambiguous seed instances (or their ambiguous contexts) before running the initial expansion algorithm. It is our hope that most of the errors identified in this work could be automatically discovered without any manual judgments.

References

Abney, S. Parsing by Chunks. 1991. In: Robert Berwick, Steven Abney and Carol Tenny (eds.), *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht.

Banko, M. and Brill, E. 2001. Scaling to very large corpora for natural language disambiguation. In *Proceedings of ACL-2001*.pp 26-33. Morristown, NJ.

Banko, M.; Cafarella, M.; Soderland, S.; Broadhead, M.; Etzioni, O. 2007. Open Information Extraction from the Web. In *Proceedings of IJCAI-07*.

Bayardo, R. J; Yiming Ma,; Ramakrishnan Srikant.; Scaling Up All-Pairs Similarity Search. In Proc. of the 16th Int'l Conf. on World Wide Web. pp 131-140 2007.

Brill, E. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. *Computational Linguistics*.

Bunescu, R. and Mooney, R. 2004 Collective Information Extraction with Relational Markov Networks. In *Proceedings of ACL-04*.pp. 438-445.

Cohn, D. A., Atlas, L., and Ladner, R. E. 1994. Improving Generalization with Active Learning. Machine Learning, 15(2):201-221. Springer, Netherlands.

Dagan, I. and Engelson, S. P. 1995. Selective Sampling in Natural Language Learning. In *Proceedings of IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing*. Montreal, Canada.

Downey, D.; Broadhead, M; Etzioni, O. 2007. Locating Complex Named Entities in Web Text. In *Proceedings of IJCAI-07.*

Etzioni, O.; Cafarella, M.; Downey. D.; Popescu, A.; Shaked, T; Soderland, S.; Weld, D.; Yates, A. 2005. Unsupervised named-entity extraction from the Web: An Experimental Study. In Artificial Intelligence, 165(1):91-134.

Harris, Z. 1985. Distributional Structure. In: Katz, J. J. (ed.), The Philosophy of Linguistics. New York: Oxford University Press. pp. 26-47.

Harman, D. 1992. Relevance feedback revisited. In *Proceeedings of SIGIR-92*. Copenhagen, Denmark.

Hearst, M. A. 1992.Automatic acquisition of hyponyms from large text corpora.In *Proceedings of COLING-92*. Nantes, France.

Kozareva, Z., Riloff, E. and Hovy, E. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs.In *Proceedings of ACL-08*.pp 1048-1056. Columbus, OH

Lin, D. 1998.Automatic retrieval and clustering of similar words.In *Proceedings of COLING/ACL-98*.pp. 768–774. Montreal, Canada.

Nadeau, D., Turney, P. D. and Matwin., S. 2006. Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity. In *Advances in Artifical Intelligence*.pp 266-277. Springer Berlin, Heidelberg.

Negri, M. 2004. Sense-based blind relevance feedback for question answering. In *Proceedings of SIGIR-04 Workshop on Information Retrieval For Question Answering (IR4QA)*. Sheffield, UK,

Pantel, P. and Lin, D. 2002. Discovering Word Senses from Text. In *Proceedings of KDD-02*.pp. 613-619. Edmonton, Canada.

Paşca, M. 2007a.Weakly-supervised discovery of named entities using web search queries. In *Proceedings of CIKM-07*.pp. 683-690.

Pasca, M. 2007b. Organizing and Searching the World Wide Web of Facts - Step Two: Harnessing the Wisdom of the Crowds. In *Proceedings of WWW-07*. pp. 101-110.

Paşca, M.; Lin, D.; Bigham, J.; Lifchits, A.; Jain, A. 2006. Names and Similarities on the Web: Fact Extraction in the Fast Lane. In *Proceedings of ACL-2006*.pp. 113-120.

Paşca, M. and Durme, B.J. 2008. Weakly-supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. In *Proceedings of ACL-08*.

Riloff, E. and Jones, R. 1999 Learning Dictionaries for Information Extraction by Multi-Level Boostrapping.In *Proceedings of AAAI/IAAAI-99*.

Riloff, E. and Shepherd, J. 1997. A corpus-based approach for building semantic lexicons.In *Proceedings of EMNLP-97*.

Sarmento, L.; Jijkuon, V.; de Rijke, M.; and Oliveira, E. 2007. "More like these": growing entity classes from seeds. In *Proceedings of CIKM-07*. pp. 959-962. Lisbon, Portugal.

Stevenson, M., Guo, Y. and Gaizauskas, R. 2008. Acquiring Sense Tagged Examples using Relevance Feedback. In *Proceedings ofCOLING-08.* Manchester UK.

Tang, M., Luo, X., and Roukos, S. 2001. Active learning for statistical natural language parsing.In *Proceedings of ACL-2001*.pp 120 -127. Philadelphia, PA.

Vishwa. V, Wood, K., Milic-Frayling, N. and Cox, I. J. 2005. Comparing Relevance Feedback Algorithms for Web Search. In *Proceedings of WWW 2005*. Chiba, Japan.

Wang. R.C. and Cohen, W.W. 2007.Language-Independent Set Expansion of Named Entities Using the Web.In *Proceedings of ICDM-07*.

Zhou, X. S. and Huang, S. T. 2003. Relevance Feedback in Image Retrieval: A Comprehensive Review - Xiang Sean Zhou, Thomas S. Huang Multimedia Systems. pp 8:536-544.

Zhou, G. and Su, J. 2001. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of ACL-2001*.pp. 473-480. Morristown, NJ.

Unsupervised Constraint Driven Learning For Transliteration Discovery

Ming-Wei Chang Dan Goldwasser Dan Roth Yuancheng Tu
University of Illinois at Urbana Champaign
Urbana, IL 61801
{mchang21,goldwas1,danr,ytu}@uiuc.edu

Abstract

This paper introduces a novel unsupervised *constraint-driven* learning algorithm for identifying named-entity (NE) transliterations in bilingual corpora. The proposed method does not require any annotated data or aligned corpora. Instead, it is bootstrapped using a simple resource – a romanization table. We show that this resource, when used in conjunction with constraints, can efficiently identify transliteration pairs. We evaluate the proposed method on transliterating English NEs to three different languages - Chinese, Russian and Hebrew. Our experiments show that constraint driven learning can significantly outperform existing unsupervised models and achieve competitive results to existing supervised models.

1 Introduction

Named entity (NE) transliteration is the process of transcribing a NE from a source language to some target language while preserving its pronunciation in the original language. Automatic NE transliteration is an important component in many cross-language applications, such as Cross-Lingual Information Retrieval (CLIR) and Machine Translation(MT) (Hermjakob et al., 2008; Klementiev and Roth, 2006a; Meng et al., 2001; Knight and Graehl, 1998).

It might initially seem that transliteration is an easy task, requiring only finding a phonetic mapping between character sets. However simply matching every source language character to its target language counterpart is not likely to work well as in practice this mapping depends on the context the characters appear in and on transliteration conventions which may change across domains. As a result, current approaches employ machine learning methods which, given enough labeled training data learn how to determine whether a pair of words constitute a transliteration pair. These methods typically require training data and language-specific expertise which may not exist for many languages. In this paper we try to overcome these difficulties and show that when the problem is modeled correctly, a simple character level mapping is a sufficient resource.

In our experiments, English was used as the source language, allowing us to use romanization tables, a resource commonly-available for many languages[1]. These tables contain an incomplete mapping between character sets, mapping every character to its most common counterpart.

Our transliteration model takes a discriminative approach. Given a word pair, the model determines if one word is a transliteration of the other. The features used by this model are character n-gram matches across the two strings. For example, Figure 1 describes the decomposition of a word pair into unigram features as a bipartite graph in which each edge represents an active feature.

We enhance the initial model with constraints, by framing the feature extraction process as a *structured prediction problem* - given a word pair, the set of possible active features is defined as a set of latent binary variables. The contextual dependency be-

[1]The romanization tables available at the Library of Congress website (http://www.loc.gov/catdir/cpso/roman.html) cover more than 150 languages written in various non-Roman scripts

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 299–307,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

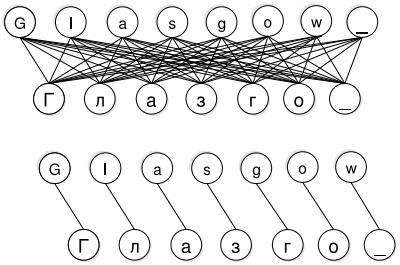


Figure 1: Top: The space of all possible features that can be generated given the word pair. Bottom: A pruned features representation generated by the inference process.

tween features is encoded as a set of constraints over these variables. Features are extracted by finding an assignment that maximizes the similarity score between the two strings and conforms to the constraints. The model is bootstrapped using a romanization table and uses a discriminatively self-trained classifier as a way to improve over several training iterations. Furthermore, when specific knowledge about the source and target languages exists, it can be directly injected into the model as constraints.

We tested our approach on three very different languages - Russian, a Slavic language, Hebrew a Semitic language, and Chinese, a Sino-Tibetan language. In all languages, using this simple resource in conjunction with constraints provided us with a robust transliteration system which significantly outperforms existing unsupervised approaches and achieves comparable performance to supervised methods.

The rest of the paper is organized as follows. Sec. 2 briefly examines more related work. Sec. 3 explains our model and Sec. 4 provide a linguistic intuition for it. Sec. 5 describes our experiments and evaluates our results followed by sec. 6 which concludes our paper.

2 Related Works

Transliteration methods typically fall into two categories: generative approaches (Li et al., 2004; Jung et al., 2000; Knight and Graehl, 1998) that try to produce the target transliteration given a source language NE, and discriminative approaches (Goldwasser and Roth, 2008b; Bergsma and Kondrak, 2007; Sproat et al., 2006; Klementiev and Roth, 2006a), that try to identify the correct transliteration for a word in the source language given several candidates in the target language. Generative methods encounter the Out-Of-Vocabulary (OOV) problem and require substantial amounts of training data and knowledge of the source and target languages. Discriminative approaches, when used to for discovering NE in a bilingual corpora avoid the OOV problem by choosing the transliteration candidates from the corpora. These methods typically make very little assumptions about the source and target languages and require considerably less data to converge. Training the transliteration model is typically done under supervised settings (Bergsma and Kondrak, 2007; Goldwasser and Roth, 2008b), or weakly supervised settings with additional temporal information (Sproat et al., 2006; Klementiev and Roth, 2006a). Our work differs from these works in that it is completely unsupervised and makes no assumptions about the training data.

Incorporating knowledge encoded as constraints into learning problems has attracted a lot of attention in the NLP community recently. This has been shown both in supervised settings (Roth and Yih, 2004; Riedel and Clarke, 2006) and unsupervised settings (Haghighi and Klein, 2006; Chang et al., 2007) in which constraints are used to bootstrap the model. (Chang et al., 2007) describes an unsupervised training of a Constrained Conditional Model (CCM), a general framework for combining statistical models with declarative constraints. We extend this work to include constraints over possible assignments to latent variables which, in turn, define the underlying representation for the learning problem.

In the transliteration community there are several works (Ristad and Yianilos, 1998; Bergsma and Kondrak, 2007; Goldwasser and Roth, 2008b) that show how the feature representation of a word pair can be restricted to facilitate learning a string similarity model. We follow the approach discussed in (Goldwasser and Roth, 2008b), which considers the feature representation as a structured prediction problem and finds the set of optimal assignments (or feature activations), under a set of legitimacy constraints. This approach stresses the importance of interaction between learning and inference, as the model iteratively uses inference to improve the sample representation for the learning problem and uses the learned model to improve the accuracy of the in-

ference process. We adapt this approach to unsupervised settings, where iterating over the data improves the model in both of these dimensions.

3 Unsupervised Constraint Driven Learning

In this section we present our Unsupervised Constraint Driven Learning (**UCDL**) model for discovering transliteration pairs. Our task is in essence a ranking task. Given a NE in the source language and a list of candidate transliterations in the target language, the model is supposed to rank the candidates and output the one with the highest score. The model is bootstrapped using two linguistic resources: a *romanization table* and a set of general and linguistic constraints. We use several iterations of self training to learn the model. The details of the procedure are explained in Algorithm 1.

In our model features are character pairs (c_s, c_t), where $c_s \in C_s$ is a source word character and $c_t \in C_t$ is a target word character. The feature representation of a word pair v_s, v_t is denoted by $F(v_s, v_t)$. Each feature (c_s, c_t) is assigned a weight $W(c_s, c_t) \in R$. In step 1 of the algorithm we initialize the weights vector using the romanization table.

Given a pair (v_s, v_t), a feature extraction process is used to determine the feature based representation of the pair. Once features are extracted to represent a pair, the sum of the weights of the extracted features is the score assigned to the target transliteration candidate. Unlike traditional feature extraction approaches, our feature representation function does not produce a fixed feature representation. In step 2.1, we formalize the feature extraction process as a constrained optimization problem that captures the interdependencies between the features used to represent the sample. That is, obtaining $F(v_s, v_t)$ requires solving an optimization problem. The technical details are described in Sec. 3.1. The constraints we use are described in Sec. 3.2.

In step 2.2 the different candidates for every source NE are ranked according to the similarity score associated with their chosen representation. This ranking is used to "label" examples for a discriminative learning process that learns increasingly better weights, and thus improve the representation of the pair: each source NE paired with its top ranked transliteration is labeled as a positive examples (step 2.3) and the rest of the samples are considered as negative samples. In order to focus the learning process, we removed from the training set all negative examples ruled-out by the constraints (step 2.4). As the learning process progresses, the initial weights are replaced by weights which are discriminatively learned (step 2.5). This process is repeated several times until the model converges, and repeats the same ranking over several iterations.

Input: Romanization table $\mathcal{T} : C_s \rightarrow C_t$, Constraints $\mathcal{C}$, Source NEs: V_s, Target words: V_t

1. Initialize Model

Let $\mathcal{W} : C_s \times C_t \rightarrow R$ be a weight vector.
Initialize $\mathcal{W}$ using $\mathcal{T}$ by the following procedure

$$\forall (c_s, c_t), (c_s, c_t) \in \mathcal{T} \Rightarrow \mathcal{W}(c_s, c_t) = 0,$$
$$\forall (c_s, c_t), \neg((c_s, c_t) \in \mathcal{T}) \Rightarrow \mathcal{W}(c_s, c_t) = -1,$$
$$\forall c_s, \mathcal{W}(c_s, \_) = -1, \forall c_t, \mathcal{W}(\_, c_t) = -1.$$

2. Constraints driven unsupervised training

while *not converged* **do**

1. $\forall v_s \in V_s, v_t \in V_t$, use $\mathcal{C}$ and $\mathcal{W}$ to generate a representation $F(v_s, v_t)$
2. $\forall v_s \in V_s$, find the top ranking transliteration pair (v_s, v_t^*) by solving $v_t^* = \arg\max_{v_t} \text{score}(F(v_s, v_t))$.
3. $D = \{(+, F(v_s, v_t^*)) \mid \forall v_s \in V_s\}$.
4. $\forall v_s \in V_s, v_t \in V_t$, if $v_t \neq v_t^*$ and $\text{score}(F(v_s, v_t)) \neq -\infty$, then $D = D \cup \{(-, F(v_s, v_t))\}$.
5. $\mathcal{W} \leftarrow \text{train}(D)$

end

Algorithm 1: UCDL Transliteration Framework.

In the rest of this section we explain this process in detail. We define the feature extraction inference process in Sec. 3.1, the constraints used in Sec. 3.2 and the inference algorithm in Sec. 3.3. The linguistic intuition for our model is described in Sec. 4.

3.1 Finding Feature Representation as Constrained Optimization

We use the formulation of Constrainted Conditional Models (CCMs) (Roth and Yih, 2004; Roth and Yih, 2007; Chang et al., 2008). Previous work on CCM models dependencies between different decisions in structured prediction problems. Transliteration discovery is a binary classification problem, however,

the underlying representation of each sample can be modeled as a CCM, defined as a set of latent variables corresponding to the set of all possible features for a given sample. The dependencies between the features are captured using constraints.

Given a word pair, the set of all possible features consists of all character mappings from the source word to the target word. Since in many cases the size of the words differ we augment each of the words with a blank character (denoted as '_'). We model character omission by mapping the character to the blank character. This process is formally defined as an operator mapping a transliteration candidate pair to a set of binary variables, denoted as $All\text{-}Features\ (AF)$.

$$AF = \{(c_s, c_t) | c_s \in v_s \cup \{\_\}, c_t \in v_t \cup \{\_\}\}$$

This representation is depicted at the top of Figure 1.

The initial sample representation (AF) generates features by coupling substrings from the two terms without considering the dependencies between the possible combinations. This representation is clearly noisy and in order to improve it we select a subset $F \subset AF$ of the possible features. The selection process is formulated as a linear optimization problem over binary variables encoding feature activations in AF. Variables assigned 1 are selected to be in F, and those assigned 0 are not. The objective function maximized is a linear function over the variables in AF, each with its weight as a coefficient, as in the left part of Equation 1 below. We seek to maximize this linear sum subject to a set of constraints. These represent the dependencies between selections and prior knowledge about possible legitimate character mappings and correspond to the right side of Equation 1. In our settings only *hard constraints* are used and therefore the penalty (ρ) for violating any of the constraints is set to ∞. The specific constraints used are discussed in Sec. 3.2. The **score** of the mapping $F(v_s, v_t)$ can be written as follows:

$$\frac{1}{|v_t|}(\mathcal{W} \cdot F(v_s, v_t) - \sum_{c_i \in \mathcal{C}} \rho c_i(F(v_s, v_t)) \qquad (1)$$

We normalize this score by dividing it by the size of the target word, since the size of candidates varies, normalization improved the ranking of candidates.

The result of the optimization process is a set F of active features, defined in Equation 2. The result of this process is described at the bottom of Figure 1.

$$F^*(v_s, v_t) = \arg\max_{F \subset AF(v_s, v_t)} \text{score}(F). \quad (2)$$

The ranking process done by our model can now be naturally defined - given a source word v_s, and a set of candidates target words $v_t^0, \ldots, v_t^n$, find the candidate whose optimal representation maximizes Equation 1. This process is defined in Equation 3.

$$v_t^* = \arg\max_{v_t^i} \text{score}(F(v_s, v_t^i)). \qquad (3)$$

3.2 Incorporating Mapping Constraints

We consider two types of constraints: language specific and general constraints that apply to all languages. Language specific constraints typically impose a local restriction such as individually forcing some of the possible character mapping decisions. The linguistic intuition behind these constraints is discussed in Section 4. General constraints encode global restrictions, capturing the dependencies between different mapping decisions.

General constraints: To facilitate readability we denote the feature mapping the i-th source word character to the j-th target word character as a Boolean variable a_{ij} that is 1 if that feature is active and 0 otherwise.

- *Coverage* - Every character must be mapped only to a single character or to the blank character. We can formulate this as: $\sum_j a_{ij} = 1$ and $\sum_i a_{ij} = 1$.
- *No Crossing* - Every character mapping, except mapping to blank character, should preserve the order of appearance in the source and target words, or formally - $\forall i, j$ s.t. $a_{ij} = 1 \Rightarrow \forall l < i, \forall k > j, a_{lk} = 0$. Another constraint is $\forall i, j$ s.t. $a_{ij} = 1 \Rightarrow \forall l > i, \forall k < j, a_{lk} = 0$.

Language specific constraints

- *Restricted Mapping*: These constraints restrict the possible local mappings between source and target language characters. We maintain a set of possible mappings $\{c_s \rightarrow \Theta_{c_s}\}$, where $\Theta_{c_s} \subseteq C_t$ and $\{c_t \rightarrow \Theta_{c_t}\}$, where $\Theta_{c_t} \subseteq C_s$. Any feature (c_s, c_t) such that $c_s \notin \Theta_{c_t}$ or $c_t \notin \Theta_{c_s}$ is penalized in our model.

- *Length restriction*: An additional constraint restricts the size difference between the two words. We formulate this as follows: $\forall v_s \in V_s, \forall v_t \in V_t$, if $\gamma|v_t| > |v_s|$ and $\gamma|v_s| > |v_t|$, score$(F(v_s, v_t)) = -\infty$. Although γ can take different values for different languages, we simply set γ to 2 in this paper.

In addition to biasing the model to choose the right candidate, the constraints also provide a computational advantage: a given a word pair is eliminated from consideration when the length restriction is not satisfied or there is no way to satisfy the restricted mapping constraints.

3.3 Inference

The optimization problem defined in Equation 2 is an integer linear program (ILP). However, given the structure of the problem it is possible to develop an efficient dynamic programming algorithm for it, based on the algorithm for finding the minimal edit distance of two strings. The complexity of finding the optimal set of features is only quadratic in the size of the input pair, a clear improvement over the ILP exponential time algorithm. The algorithm minimizes the weighted edit distance between the strings, and produces a character alignment that satisfies the general constraints (Sec. 3.2). Our modifications are only concerned with incorporating the language-specific constraints into the algorithm. This can be done simply by assigning a negative infinity score to any alignment decision not satisfying these constraints.

4 Bootstrapping with Linguistic Information

Our model is bootstrapped using two resources - a romanization table and mapping constraints. Both resources capture the same information - character mapping between languages. The distinction between the two represents the difference in the confidence we have in these resources - the romanization table is a noisy mapping covering the character set and is therefore better suited as a feature. Constraints, represented by pervasive, correct character mapping, indicate the sound mapping tendency between source and target languages. For example, certain n-gram phonemic mappings, such as $r \rightarrow l$ from English to Chinese, are language specific and can be captured by language specific sound change patterns.

Phonemes	Constraints
Vowel	$i \rightarrow y;\ u \rightarrow w; a \rightarrow a$
Nasal	$m \leftrightarrow m;\ \mathbf{m, n} \leftarrow \mathbf{m}$
Approximant	$r \rightarrow l;\ \mathbf{l, r} \leftrightarrow \mathbf{l}$ $l \leftarrow l;\ w \rightarrow h, w, f$ $h, o, u, v \leftarrow w;\ y \rightarrow y$
Fricative	$v \rightarrow w, b, f$ $s \rightarrow s, x, z;\ s, c \leftarrow s$
Plosive	$p \rightarrow b, p;\ p \leftarrow p$ $b \rightarrow b;\ t \leftarrow t$ $t, d \leftarrow d;\ q \rightarrow k$

Table 1: All language specific constraints used in our English to Chinese transliteration (see Sec. 3.2 for more details). Constraints in boldface apply to all positions, the rest apply only to characters appearing in initial position.

These patterns have been used by other systems as features or *pseudofeatures* (Yoon et al., 2007). However, in our system these language specific rule-of-thumbs are systematically used as constraints to exclude impossible alignments and therefore generate better features for learning. We listed in Table 1 all 20 language specific constraints we used for Chinese. There is a total of 24 constraints for Hebrew and 17 for Russian.

The constraints in Table 1 indicate a systematic sound mapping between English and Chinese unigram character mappings. Arranged by manners of articulation each row of the table indicates the sound change tendency among vowels, nasals, approximants (retroflex and glides), fricatives and plosives. For example, voiceless plosive sounds such as *p, t* in English, tend to map to both voiced (such as *b, d*) and voiceless sounds in Chinese. However, if the sound is voiceless in Chinese, its backtrack English sound must be voiceless. This voice-voiceless sound change tendency is captured by our constraints such as $p \rightarrow b, p$ and $p \leftarrow p;\ t \leftarrow t$.

5 Experiments and Analysis

In this section, we demonstrate the effectiveness of constraint driven learning empirically. We start by describing the datasets and experimental settings and then proceed to describe the results. We evaluated our method on three very different target lan-

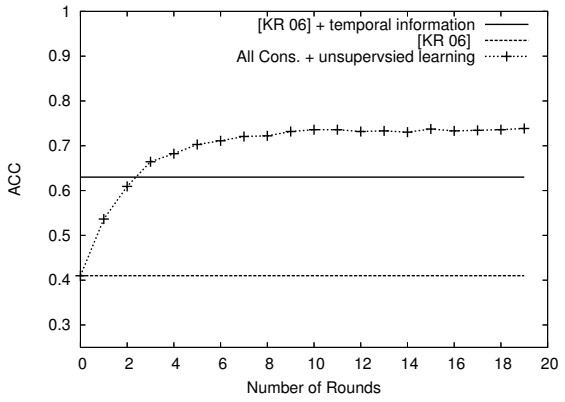


Figure 2: **Comparison between our models and weakly supervised learning methods (Klementiev and Roth, 2006b).** Note that one of the models proposed in (Klementiev and Roth, 2006b) takes advantage of the temporal information. Our best model, the unsupervised learning with all constraints, outperforms both models in (Klementiev and Roth, 2006b), even though we do not use any temporal information.

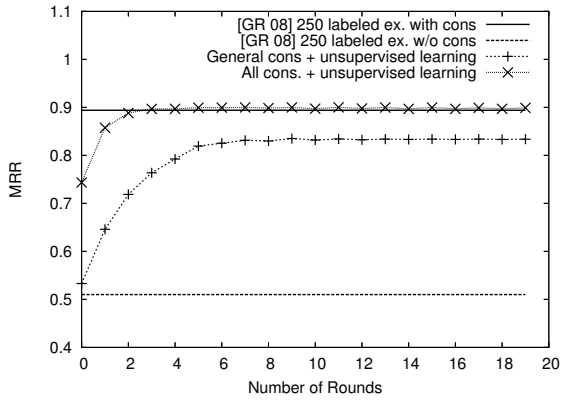


Figure 3: **Comparison between our works and supervised models in (Goldwasser and Roth, 2008b).** We show the learning curves for Hebrew under two different settings: unsupervised learning with general and all constraints. The results of two *supervised models* (Goldwasser and Roth, 2008b) are also included here. Note that our *unsupervised* model with all constraints is competitive to the supervised model with 250 labeled examples. See the text for more comparisons and details.

guages: *Russian, Chinese, and Hebrew*, and compared our results to previously published results.

5.1 Experimental Settings

In our experiments the system is evaluated on its ability to correctly identify the gold transliteration for each source word. We evaluated the system's performance using two measures adopted in many transliteration works. The first one is Mean Reciprocal Rank (MRR), used in (Tao et al., 2006; Sproat et al., 2006), which is the average of the multiplicative inverse of the rank of the correct answer. Formally, Let n be the number of source NEs. Let GoldRank(i) be the rank the algorithm assigns to the correct transliteration. Then, MRR is defined by:

$$\text{MRR} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\text{goldRank}(i)}.$$

Another measure is Accuracy (ACC) used in (Klementiev and Roth, 2006a; Goldwasser and Roth, 2008a), which is the percentage of the top rank candidates being the gold transliteration. In our implementation we used the support vector machine (SVM) learning algorithm with linear kernel as our underlying learning algorithm (mentioned in part 2.5 of Algorithm 1) . We used the package **LIBLINEAR** (Hsieh et al., 2008) in our experiments. Through all of our experiments, we used the 2-norm hinge loss as our loss function and fixed the regularization parameter C to be 0.5.

5.2 Datasets

We experimented using three different target languages *Russian, Chinese, and Hebrew*. We used English as the source language in all these experiments.

The Russian data set[2], originally introduced in (Klementiev and Roth, 2006b), is comprised of temporally aligned news articles. The dataset contains 727 single word English NEs with a corresponding set of 50,648 potential Russian candidate words which include not only name entities, but also other words appearing in the news articles.

The Chinese dataset is taken directly from an English-Chinese transliteration dictionary, derived from LDC Gigaword corpus[3]. The entire dictionary consists of 74,396 pairs of English-Chinese NEs, where Chinese NEs are written in *Pinyin*, a romanized spelling system of Chinese. In (Tao et al., 2006) a dataset which contains about 600 English NEs and 700 Chinese candidates is used. Since the dataset is not publicly available, we created a dataset in a similar way. We randomly selected approximately 600 NE pairs and then added about 100 candidates which do not correspond to any of the English NE

[2]The corpus is available http://L2R.cs.uiuc.edu/∼cogcomp.
[3]http://www.ldc.upenn.edu

Language	UCDL	Prev. works
Rus. (ACC)	73	63 (41) (KR'06)
Heb. (MRR)	0.899	0.894 (GR'08)

Table 2: Comparison to previously published results. UCDL is our method, KR'06 is described in (Klementiev and Roth, 2006b) and GR'08 in (Goldwasser and Roth, 2008b). Note that our results for Hebrew are comparable with a supervised system.

previously selected.

The Hebrew dataset, originally introduced in (Goldwasser and Roth, 2008a), consists of 300 English-Hebrew pairs extracted from Wikipedia.

5.3 Results

We begin by comparing our model to previously published models tested over the same data, in two different languages, Russian and Hebrew. For Russian, we compare to the model presented in (Klementiev and Roth, 2006b), a weakly supervised algorithm that uses both phonetic information and temporal information. The model is bootstrapped using a set of 20 labeled examples. In their setting the candidates are ranked by combining two scores, one obtained using the transliteration model and a second by comparing the relative occurrence frequency of terms over time in both languages. Due to computational tractability reasons we slightly changed Algorithm 1 to use only a small subset of the possible negative examples.

For Hebrew, we compare to the model presented in (Goldwasser and Roth, 2008b), a supervised model trained using 250 labeled examples. This model uses a bigram model to represent the transliteration samples (i.e., features are generated by pairing character unigrams and bigrams). The model also uses constraints to restrict the feature extraction process, which are equivalent to the *coverage* constraint we described in Sec. 3.2.

The results of these experiments are reported using the evaluation measures used in the original papers and are summarized in Table 2. The results show a significant improvement over the Russian data set and comparable performance to the supervised method used for Hebrew.

Figure 2 describes the learning curve of our method over the Russian dataset. We compared our algorithm to two models described in (Klementiev and Roth, 2006b) - one uses only phonetic similarity and the second also considers temporal co-occurrence similarity when ranking the transliteration candidates. Both models converge after 50 iterations. When comparing our model to their models, we found that even though our model ignores the temporal information it achieves better results and converges after fewer iterations. Their results report a significant improvement when using temporal information - improving an ACC score of 41% without temporal information to 63% when using it. Since the temporal information is orthogonal to the transliteration model, our model should similarly benefit from incorporating the temporal information.

Figure 3 compares the learning curve of our method to an existing supervised method over the Hebrew data and shows we get comparable results.

Unfortunately, we could not find a published Chinese dataset. However, our system achieved similar results to other systems, over a different dataset with similar number of training examples. For example, (Sproat et al., 2006) presents a supervised system that achieves a MRR score of 0.89, when evaluated over a dataset consisting of 400 English NE and 627 Chinese words. Our results for a different dataset of similar size are reported in Table 3.

5.4 Analysis

The resources used in our framework consist of - a romanization table, general and language specific transliteration constraints. To reveal the impact of each component we experimented with different combination of the components, resulting in three different testing configurations.

Romanization Table: We initialized the weight vector using a romanization table and did not use any constraints. To generate features we use a modified version of our AF operator (see Sec. 3), which generates features by coupling characters in close positions in the source and target words. This configuration is equivalent to the model used in (Klementiev and Roth, 2006b).

+General Constraints: This configuration uses the romanization table for initializing the weight vector and general transliteration constraints (see Sec. 3.2) for feature extraction.

+All Constraints: This configuration uses language specific constraints in addition to the gen-

Settings		Chinese	Russian	Hebrew
Romanization table		0.019 (0.5)	0.034 (1.0)	0.046 (1.7)
Romanization table	+learning	0.020 (0.3)	0.048 (1.3)	0.028 (0.7)
+Gen Constraints		0.746 (67.1)	0.809 (74.3)	0.533 (45.0)
+Gen Constraints	+learning	0.867 (82.2)	0.906 (86.7)	0.834 (76.0)
+All Constraints		0.801 (73.4)	0.849 (79.3)	0.743 (66.0)
+All Constraints	+learning	**0.889 (84.7)**	**0.931 (90.0)**	**0.899 (85.0)**

Table 3: Results of an ablation study of unsupervised method for three target languages. Results for ACC are *inside* parentheses, and for MRR *outside*. When the learning algorithm is used, the results after 20 rounds of constraint driven learning are reported. Note that using linguistic constraints has a significant impact in the English-Hebrew experiments. Our results show that a small amount of constraints can go a long way, and better constraints lead to better learning performance.

eral transliteration constraints to generate the feature representation. (see Sec. 4).

+Learning: Indicates that after initializing the weight vector, we update the weight using Algorithm 1. In all of the experiments, we report the results after 20 training iterations.

The results are summarized in Table 3. Due to the size of the Russian dataset, we used a subset consisting of 300 English NEs and their matching Russian transliterations for the analysis presented here. After observing the results, we discovered the following regularities for all three languages. Using the romanization table directly without constraints results in very poor performance, even after learning. This can be used as an indication of the difficulty of the transliteration problem and the difficulties earlier works have had when using only romanization tables, however, when used in conjunction with constraints results improve dramatically. For example, in the English-Chinese data set, we improve MRR from 0.02 to 0.746 and for the English-Russian data set we improve 0.03 to 0.8. Interestingly, the results for the English-Hebrew data set are lower than for other languages - we achieve 0.53 MRR in this setting. We attribute the difference to the quality of the mapping in the romanization table for that language. Indeed, the weights learned after 20 training iterations improve the results to 0.83. This improvement is consistent across all languages, after learning we are able to achieve a MRR score of 0.87 for the English-Chinese data set and 0.91 for the English-Russian data set. These results show that romanization table contains enough information to bootstrap the model when used in conjunction with constraints. We are able to achieve results comparable to supervised methods that use a similar set of constraints and labeled examples.

Bootstrapping the weight vector using language specific constraints can further improve the results. They provide several advantages: a better starting point, an improved learning rate and a better final model. This is clear in all three languages, for example results for the Russian and Chinese bootstrapped models improve by 5%, and by over 20% for Hebrew. After training the difference is smaller- only 3% for the first two and 6% for Hebrew. Figure 3 describes the learning curve for models with and without language specific constraints for the English-Hebrew data set, it can be observed that using these constraints the model converges faster and achieves better results.

6 Conclusion

In this paper we develop a constraints driven approach to named entity transliteration. In doing it we show that romanization tables are a very useful resource for transliteration discovery if proper constraints are included. Our framework does not need labeled data and does not assume that bilingual corpus are temporally aligned. Even without using any labeled data, our model is competitive to existing supervised models and outperforms existing weakly supervised models.

7 Acknowledgments

We wish to thank the reviewers for their insightful comments. This work is partly supported by NSF grant SoD-HCER-0613885 and DARPA funding under the Bootstrap Learning Program.

References

S. Bergsma and G. Kondrak. 2007. Alignment-based discriminative string similarity. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 656–663, Prague, Czech Republic, June. Association for Computational Linguistics.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 280–287, Prague, Czech Republic, Jun. Association for Computational Linguistics.

M. Chang, L. Ratinov, N. Rizzolo, and D. Roth. 2008. Learning and inference with constraints. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, July.

D. Goldwasser and D. Roth. 2008a. Active sample selection for named entity transliteration. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, June.

D. Goldwasser and D. Roth. 2008b. Transliteration as constrained optimization. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 353–362, Oct.

A. Haghighi and D. Klein. 2006. Prototype-driven learning for sequence models. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

U. Hermjakob, K. Knight, and H. Daumé III. 2008. Name translation in statistical machine translation - learning when to transliterate. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 389–397, Columbus, Ohio, June.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear svm. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 408–415, New York, NY, USA. ACM.

S. Jung, S. Hong, and E. Paek. 2000. An english to korean transliteration model of extended markov window. In *Proc. the International Conference on Computational Linguistics (COLING)*, pages 383–389.

A. Klementiev and D. Roth. 2006a. Named entity transliteration and discovery from multilingual comparable corpora. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 82–88, June.

A. Klementiev and D. Roth. 2006b. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages USS,TL,ADAPT, July.

K. Knight and J. Graehl. 1998. Machine transliteration. *Computational Linguistics*, pages 599–612.

H. Li, M. Zhang, and J. Su. 2004. A joint source-channel model for machine transliteration. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 159–166, Barcelona, Spain, July.

H. Meng, W. Lo, B. Chen, and K. Tang. 2001. Generating phonetic cognates to handle named entities in english-chinese cross-langauge spoken document retreival. In *Proceedings of the Automatic Speech Recognition and Understanding Workshop*, pages 389–397.

S. Riedel and J. Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 129–137, Sydney, Australia.

E. S. Ristad and P. N. Yianilos. 1998. Learning string edit distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(5):522–532, May.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. pages 1–8. Association for Computational Linguistics.

D. Roth and W. Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

R. Sproat, T. Tao, and C. Zhai. 2006. Named entity transliteration with comparable corpora. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 73–80, Sydney, Australia, July.

T. Tao, S. Yoon, A. Fister, R. Sproat, and C. Zhai. 2006. Unsupervised named entitly transliteration using temporal and phonetic correlation. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 250–257.

S. Yoon, K. Kim, and R. Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 112–119, Prague, Czech Republic, June.

On the Syllabification of Phonemes

Susan Bartlett[†] and **Grzegorz Kondrak**[†] and **Colin Cherry**[‡]

[†]Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8, Canada
{susan,kondrak}@cs.ualberta.ca

[‡]Microsoft Research
One Microsoft Way
Redmond, WA, 98052
colinc@microsoft.com

Abstract

Syllables play an important role in speech synthesis and recognition. We present several different approaches to the syllabification of phonemes. We investigate approaches based on linguistic theories of syllabification, as well as a discriminative learning technique that combines Support Vector Machine and Hidden Markov Model technologies. Our experiments on English, Dutch and German demonstrate that our transparent implementation of the sonority sequencing principle is more accurate than previous implementations, and that our language-independent SVM-based approach advances the current state-of-the-art, achieving word accuracy of over 98% in English and 99% in German and Dutch.

1 Introduction

Syllabification is the process of dividing a word into its constituent syllables. Although some work has been done on syllabifying orthographic forms (Müller et al., 2000; Bouma, 2002; Marchand and Damper, 2007; Bartlett et al., 2008), syllables are, technically speaking, phonological entities that can only be composed of strings of phonemes. Most linguists view syllables as an important unit of prosody because many phonological rules and constraints apply within syllables or at syllable boundaries (Blevins, 1995).

Apart from their purely linguistic significance, syllables play an important role in speech synthesis and recognition (Kiraz and Möbius, 1998; Pearson et al., 2000). The pronunciation of a given phoneme tends to vary depending on its location within a syllable. While actual implementations vary, text-to-speech (TTS) systems must have, at minimum, three components (Damper, 2001): a letter-to-phoneme (L2P) module, a prosody module, and a synthesis module. Syllabification can play a role in all three modules.

Because of the productive nature of language, a dictionary look-up process for syllabification is inadequate. No dictionary can ever contain all possible words in a language. For this reason, it is necessary to develop systems that can automatically syllabify out-of-dictionary words.

In this paper, we advance the state-of-the-art in both categorical (non-statistical) and supervised syllabification. We outline three categorical approaches based on common linguistic theories of syllabification. We demonstrate that when implemented carefully, such approaches can be very effective, approaching supervised performance. We also present a data-driven, discriminative solution: a Support Vector Machine Hidden Markov Model (SVM-HMM), which tags each phoneme with its syllabic role. Given enough data, the SVM-HMM achieves impressive accuracy thanks to its ability to capture context-dependent generalizations, while also memorizing inevitable exceptions. Our experiments on English, Dutch and German demonstrate that our SVM-HMM approach substantially outperforms the existing state-of-the-art learning approaches. Although direct comparisons are difficult, our system achieves over 99% word accuracy on German and Dutch, and the highest reported accuracy on English.

The paper is organized as follows. We outline common linguistic theories of syllabification in Section 2, and we survey previous computational sys-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 308–316,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

tems in Section 3. Our linguistically-motivated approaches are described in Section 4. In Section 5, we describe our system based on the SVM-HMM. The experimental results are presented in Section 6.

2 Theories of Syllabification

There is some debate as to the exact structure of a syllable. However, phonologists are in general agreement that a syllable consists of a nucleus (vowel sound), preceded by an optional onset and followed by an optional coda. In many languages, both the onset and the coda can be complex, *i.e.*, composed of more than one consonant. For example, the word *breakfast* [brɛk-fəst] contains two syllables, of which the first has a complex onset [br], and the second a complex coda [st]. Languages differ with respect to various typological parameters, such as optionality of onsets, admissibility of codas, and the allowed complexity of the syllable constituents. For example, onsets are required in German, while Spanish prohibits complex codas.

There are a number of theories of syllabification; we present three of the most prevalent. The **Legality Principle** constrains the segments that can begin and end syllables to those that appear at the beginning and end of words. In other words, a syllable is not allowed to begin with a consonant cluster that is not found at the beginning of some word, or end with a cluster that is not found at the end of some word (Goslin and Frauenfelder, 2001). Thus, a word like *admit* [ədmɪt] must be syllabified [ədmɪt] because [dm] never appears word-initially or word-finally in English. A shortcoming of the legality principle is that it does not always imply a unique syllabification. For example, in a word like *askew* [əskju], the principle does not rule out any of [ə-skju], [əs-kju], or [əsk-ju], as all three employ legal onsets and codas.

The **Sonority Sequencing Principle** (SSP) provides a stricter definition of legality. The sonority of a sound is its inherent loudness, holding factors like pitch and duration constant (Crystal, 2003). Low vowels like [a], the most sonorous sounds, are high on the sonority scale, while plosive consonants like [t] are at the bottom. When syllabifying a word, SSP states that sonority should increase from the first phoneme of the onset to the syllable's nucleus, and then fall off to the coda (Selkirk, 1984). Consequently, in a word like *vintage* [vɪntɪʤ], we can rule out a syllabification like [vɪ-ntɪʤ] because [n] is more sonorant than [t]. However, SSP does not tell us whether to prefer [vɪn-tɪʤ] or [vɪnt-ɪʤ]. Moreover, when syllabifying a word like *vintner* [vɪntnər], the theory allows both [vɪn-tnər] and [vɪnt-nər], even though [tn] is an illegal onset in English.

Both the Legality Principle and SSP tell us which onsets and codas are permitted in legal syllables, and which are not. However, neither theory gives us any guidance when deciding between legal onsets. The **Maximal Onset Principle** addresses this by stating we should extend a syllable's onset at the expense of the preceding syllable's coda whenever it is legal to do so (Kahn, 1976). For example, the principle gives preference to [ə-skju] and [vɪn-tɪʤ] over their alternatives.

3 Previous Computational Approaches

Unlike tasks such as part of speech tagging or syntactic parsing, syllabification does not involve structural ambiguity. It is generally believed that syllable structure is usually predictable in a language provided that the rules have access to all conditioning factors: stress, morphological boundaries, part of speech, etymology, etc. (Blevins, 1995). However, in speech applications, the phonemic transcription of a word is often the only linguistic information available to the system. This is the common assumption underlying a number of computational approaches that have been proposed for the syllabification of phonemes.

Daelemans and van den Bosch (1992) present one of the earliest systems on automatic syllabification: a neural network-based implementation for Dutch. Daelemans et al. (1997) also explore the application of exemplar-based generalization (EBG), sometimes called instance-based learning. EBG generally performs a simple database look-up to syllabify a test pattern, choosing the most common syllabification. In cases where the test pattern is not found in the database, the most similar pattern is used to syllabify the test pattern.

Hidden Markov Models (HMMs) are another popular approach to syllabification. Krenn (1997) introduces the idea of treating syllabification as a

tagging task. Working from a list of syllabified phoneme strings, she automatically generates tags for each phone. She uses a second-order HMM to predict sequences of tags; syllable boundaries can be trivially recovered from the tags. Demberg (2006) applies a fourth-order HMM to the syllabification task, as a component of a larger German text-to-speech system. Schmid et al. (2007) improve on Demberg's results by applying a fifth-order HMM that conditions on both the previous tags and their corresponding phonemes.

Kiraz and Möbius (1998) present a weighted finite-state-based approach to syllabification. Their language-independent method builds an automaton for each of onsets, nuclei, and codas, by counting occurrences in training data. These automatons are then composed into a transducer accepting sequences of one or more syllables. They do not report quantitative results for their method.

Pearson et al. (2000) compare two rule-based systems (they do not elaborate on the rules employed) with a CART decision tree-based approach and a "global statistics" algorithm. The global statistics method is based on counts of consonant clusters in contexts such as word boundaries, short vowels, or long vowels. Each test word has syllable boundaries placed according to the most likely location given a cluster and its context. In experiments performed with their in-house dataset, their statistics-based method outperforms the decision-tree approach and the two rule-based methods.

Müller (2001) presents a hybrid of a categorical and data-driven approach. First, she manually constructs a context-free grammar of possible syllables. This grammar is then made probabilistic using counts obtained from training data. Müller (2006) attempts to make her method language-independent. Rather than hand-crafting her context-free grammar, she automatically generates all possible onsets, nuclei, and codas, based on the phonemes existing in the language. The results are somewhat lower than in (Müller, 2001), but the approach can be more easily ported across languages.

Goldwater and Johnson (2005) also explore using EM to learn the structure of English and German phonemes in an unsupervised setting, following Müller in modeling syllable structure with PCFGs. They initialize their parameters using a deterministic parser implementing the sonority principle and estimate the parameters for their maximum likelihood approach using EM.

Marchand et al. (2007) apply their Syllabification by Analogy (SbA) technique, originally developed for orthographic forms, to the pronunciation domain. For each input word, SbA finds the most similar substrings in a lexicon of syllabified phoneme strings and then applies the dictionary syllabifications to the input word. Their survey paper also includes comparisons with a method broadly based on the legality principle. The authors find their legality-based implementation fares significantly worse than SbA.

4 Categorical Approaches

Categorical approaches to syllabification are appealing because they are efficient and linguistically intuitive. In addition, they require little or no syllable-annotated data. We present three *categorical* algorithms that implement the linguistic insights outlined in Section 2. All three can be viewed as variations on the basic pseudo-code shown in Figure 1. Every vowel is labeled as a nucleus, and every consonant is labeled as either an onset or a coda. The algorithm labels all consonants as onsets unless it is illegal to do so. Given the labels, it is straightforward to syllabify a word. The three methods differ in how they determine a "legal" onset.

As a rough baseline, the MAXONSET implementation considers all combinations of consonants to be legal onsets. Only word-final consonants are labeled as codas.

LEGALITY combines the Legality Principle with onset maximization. In our implementation, we collect all word-initial consonant clusters from the corpus and deem them to be legal onsets. With this method, no syllable can have an onset that does not appear word-initially in the training data. We do not test for the legality of codas. The performance of LEGALITY depends on the number of phonetic transcriptions that are available, but the transcriptions need not be annotated with syllable breaks.

SONORITY combines the Sonority Sequencing Principle with onset maximization. In this approach, an onset is considered legal if every member of the onset ranks lower on the sonority scale than ensuing

```
until current phoneme is a vowel
    label current phoneme as an onset
end loop
until all phonemes have been labeled
    label current phoneme as a nucleus
    if there are no more vowels in the word
        label all remaining consonants as codas
    else
        onset := all consonants before next vowel
        coda := empty
        until onset is legal
            coda := coda plus first phoneme of onset
            onset := onset less first phoneme
        end loop
    end if
end loop
Insert syllable boundaries before onsets
```

Figure 1: Pseudo-code for syllabifying a string of phonemes.

consonants. SONORITY requires no training data because it implements a sound linguistic theory. However, an existing development set for a given language can help with defining and validating additional language-specific constraints.

Several sonority scales of varying complexity have been proposed. For example, Selkirk (1984) specifies a hierarchy of eleven distinct levels. We adopt a minimalistic scale shown in Figure 2. which avoids most of the disputed sonority contrasts (Jany et al., 2007). We set the sonority distance parameter to 2, which ensures that adjacent consonants in the onset differ by at least two levels of the scale. For example, [pr] is an acceptable onset because it is composed of an obstruent and a liquid, but [pn] is not, because nasals directly follow obstruents on our sonority scale.

In addition, we incorporate several English-specific constraints listed by Kenstowicz (1994, pages 257–258). The constraints, or *filters*, prohibit complex onsets containing:

(i) two labials (e.g., [pw], [bw], [fw], [vw]),

(ii) a non-strident coronal followed by a lateral (e.g., [tl], [dl], [θl])

(iii) a voiced fricative (e.g., [vr], [zw], except [vj]),

(iv) a palatal consonant (e.g., [ʃl], [ʧr], except [ʃr]).

Sound	**Examples**	**Level**
Vowels	u, ə, ...	4
Glides	w, j, ...	3
Liquids	l, r, ...	2
Nasals	m, ŋ, ...	1
Obstruents	g, θ, ...	0

Figure 2: The sonority scale employed by SONORITY.

A special provision allows for prepending the phoneme [s] to onsets beginning with a voiceless plosive. This reflects the special status of [s] in English, where onsets like [sk] and [sp] are legal even though the sonority is not strictly increasing.

5 Supervised Approach: SVM-HMM

If annotated data is available, a classifier can be trained to predict the syllable breaks. A Support Vector Machine (SVM) is a discriminative supervised learning technique that allows for a rich feature representation of the input space. In principle, we could use a multi-class SVM to classify each phoneme according to its position in a syllable on the basis of a set of features. However, a traditional SVM would treat each phoneme in a word as an independent instance, preventing us from considering interactions between labels. In order to overcome this shortcoming, we employ an SVM-HMM[1] (Altun et al., 2003), an instance of the Structured SVM formalism (Tsochantaridis et al., 2004) that has been specialized for sequence tagging.

When training a structured SVM, each training instance x_i is paired with its label y_i, drawn from the set of possible labels, Y_i. In our case, the training instances x_i are words, represented as sequences of phonemes, and their labels y_i are syllabifications, represented as sequences of onset/nucleus/coda tags. For each training example, a feature vector $\Psi(x, y)$ represents the relationship between the example and a candidate tag sequence. The SVM finds a weight vector w, such that $w \cdot \Psi(x, y)$ separates correct taggings from incorrect taggings by as large a margin as possible. Hamming distance D_H is used to capture how close a wrong sequence y is to y_i, which

[1] http://svmlight.joachims.org/svm_struct.html

in turn impacts the required margin. Tag sequences that share fewer tags in common with the correct sequence are separated by a larger margin.

Mathematically, a (simplified) statement of the SVM learning objective is:

$$\forall_i \forall_{y \in Y_i, y \neq y_i} : \\ [\Psi(x_i, y_i) \cdot w > \Psi(x_i, y) \cdot w + D_H(y, y_i)] \quad (1)$$

This objective is only satisfied when w tags all training examples correctly. In practice, slack variables are introduced, which allow us to trade off training accuracy and the complexity of w via a cost parameter. We tune this parameter on our development set.

The SVM-HMM training procedure repeatedly uses the Viterbi algorithm to find, for the current w and each (x_i, y_i) training pair, the sequence y that most drastically violates the inequality shown in Equation 1. These incorrect tag sequences are added to a growing set, which constrains the quadratic optimization procedure used to find the next w. The process iterates until no new violating sequences are found, producing an approximation to the inequality over all $y \in Y_i$. A complete explanation is given by Tsochantaridis et al. (2004).

Given a weight vector w, a structured SVM tags new instances x according to:

$$\text{argmax}_{y \in Y} [\Psi(x, y) \cdot w] \quad (2)$$

The SVM-HMM gets the HMM portion of its name from its use of the HMM Viterbi algorithm to solve this argmax.

5.1 Features

We investigated several tagging schemes, described in detail by Bartlett (2007). During development, we found that tagging each phoneme with its syllabic role (Krenn, 1997) works better than the simple binary distinction between syllable-final and other phonemes (van den Bosch, 1997). We also discovered that accuracy can be improved by numbering the tags. Therefore, in our tagging scheme, the single-syllable word *strengths* [strɛŋθs] would be labeled with the sequence {O1 O2 O3 N1 C1 C2 C3}.

Through the use of the Viterbi algorithm, our feature vector $\Psi(x, y)$ is naturally divided into emission and transition features. Emission features link an aspect of the input word x with a single tag in the sequence y. Unlike a generative HMM, these emission features do not require any conditional independence assumptions. Transition features link tags to tags. Our only transition features are counts of adjacent tag pairs occurring in y.

For the emission features, we use the current phoneme and a fixed-size context window of surrounding phonemes. Thus, the features for the phoneme [k] in *hockey* [hɑki] might include the [ɑ] preceding it, and the [i] following it. In experiments on our development set, we found that the optimal window size is nine: four phonemes on either side of the focus phoneme. Because the SVM-HMM is a linear classifier, we need to explicitly state any important conjunctions of features. This allows us to capture more complex patterns in the language that unigrams alone cannot describe. For example, the bigram [ps] is illegal as an onset in English, but perfectly reasonable as a coda. Experiments on the development set showed that performance peaked using all unigrams, bigrams, trigrams, and four-grams found within our context window.

Method	English
MAXONSET	61.38
LEGALITY	93.16
SONORITY	95.00
SVM-HMM	98.86
tsylb	93.72

Table 1: Word accuracy on the CELEX dataset.

6 Syllabification Experiments

We developed our approach using the English portion of the CELEX lexical database (Baayen et al., 1995). CELEX provides the phonemes of a word and its correct syllabification. It does not designate the phonemes as onsets, nuclei, or codas, which is the labeling we want to predict. Fortunately, extracting the labels from a syllabified word is straightforward. All vowel phones are assigned to be nuclei; consonants preceding the nucleus in a syllable are assigned to be onsets, while consonants following the nucleus in a syllable are assigned to be codas.

The results in Table 1 were obtained on a test set of 5K randomly selected words. For training the SVM-HMM, we randomly selected 30K words not

appearing in the test set, while 6K training examples were held out for development testing. We report the performance in terms of word accuracy (entire words syllabified correctly). Among the categorical approaches, SONORITY clearly outperforms not only LEGALITY, but also *tsylb* (Fisher, 1996), an implementation of the complex algorithm of Kahn (1976), which makes use of lists of legal English onsets. Overall, our SVM-based approach is a clear winner.

The results of our discriminative method compares favorably with the results of competing approaches on English CELEX. Since there are no standard train-test splits for syllabification, the comparison is necessarily indirect, but note that our training set is substantially smaller. For her language-independent PCFG-based approach, Müller (2006) reports 92.64% word accuracy on the set of 64K examples from CELEX using 10-fold cross-validation. The Learned EBG approach of van den Bosch (1997) achieves 97.78% word accuracy when training on approximately 60K examples. Therefore, our results represent a nearly 50% reduction of the error rate.

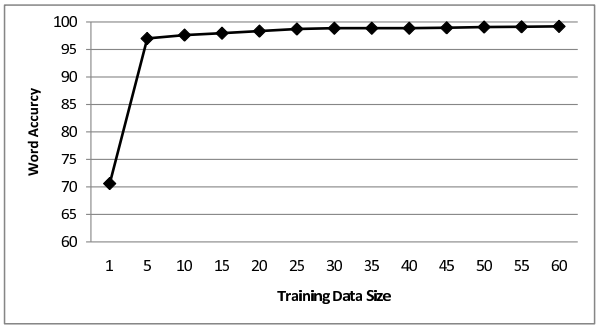


Figure 3: Word accuracy on English CELEX as a function of the number of thousands of training examples.

Though the SVM-HMM's training data requirements are lower than previous supervised syllabification approaches, they are still substantial. Figure 3 shows a learning curve over varying amounts of training data. Performance does not reach acceptable levels until 5K training examples are provided.

6.1 Error Analysis

There is a fair amount of overlap in the errors made by the SVM-HMM and the SONORITY. Table 4 shows a few characteristic examples. The CELEX syllabifications of *tooth-ache* and *pass-ports* follow the morphological boundaries of the compound words. Morphological factors are a source of errors for both approaches, but significantly more so for SONORITY. The performance difference comes mainly from the SVM's ability to handle many of these morphological exceptions. The SVM generates the correct syllabification of *northeast* [norθ-ist], even though an onset of [θ] is perfectly legal. On the other hand, the SVM sometimes overgeneralizes, as in the last example in Table 4.

SVM-HMM	SONORITY	
tu-θek	tu-θek	*toothache*
pae-sports	pae-sports	*passports*
norθ-ist	nor-θist	*northeast*
dɪs-plizd	dɪ-splizd	*displeased*
dɪs-koz	**dɪ-skoz**	*discos*

Figure 4: Examples of syllabification errors. (Correct syllabifications are shown in bold.)

6.2 The NETtalk Dataset

Marchand et al. (2007) report a disappointing word accuracy of 54.14% for their legality-based implementation, which does not accord with the results of our categorical approaches on English CELEX. Consequently, we also apply our methods to the dataset they used for their experiments: the NETtalk dictionary. NETtalk contains 20K English words; in the experiments reported here, we use 13K training examples and 7K test words.

As is apparent from Table 2, our performance degrades significantly when switching to NETtalk. The steep decline found in the categorical methods is particularly notable, and indicates significant divergence between the syllabifications employed in the two datasets. Phonologists do not always agree on the correct syllable breaks for a word, but the NETtalk syllabifications are often at odds with linguistic intuitions. We randomly selected 50 words and compared their syllabifications against those found in Merriam-Webster Online. We found that CELEX syllabifications agree with Merriam-Webster in 84% of cases, while NETtalk only agrees 52% of the time.

Figure 5 shows several words from the NETtalk

Method	English
MAXONSET	33.64
SONORITY	52.80
LEGALITY	53.08
SVM-HMM	92.99

Table 2: Word accuracy on the NETtalk dataset.

and CELEX datasets. We see that CELEX follows the maximal onset principle consistently, while NETtalk does in some instances but not others. We also note that there are a number of NETtalk syllabifications that are clearly wrong, such as the last two examples in Figure 5. The variability of NETtalk is much more difficult to capture with any kind of principled approach. Thus, we argue that low performance on NETtalk indicate inconsistent syllabifications within that dataset, rather than any actual deficiency of the methods.

NETtalk	**CELEX**	
ʧaes-taɪz	ʧae-staɪz	*chastise*
rɛz-ɪd-ənz	rɛ-zɪ-dənz	*residence*
dɪ-strɔɪ	dɪ-strɔɪ	*destroy*
fo-tɑn	fo-tɑn	*photon*
ɑr-pɛʤ-io	ɑr-pɛ-ʤi-o	*arpeggio*
ðer-ə-baʊ-t	ðɛ-rə-baʊt	*thereabout*

Figure 5: Examples of CELEX and NETtalk syllabifications.

NETtalk's variable syllabification practices notwithstanding, the SVM-HMM approach still outperforms the previous benchmark on the dataset. Marchand et al. (2007) report 88.53% word accuracy for their SbA technique using leave-one-out testing on the entire NETtalk set (20K words). With fewer training examples, we reduce the error rate by almost 40%.

6.3 Other Languages

We performed experiments on German and Dutch, the two other languages available in the CELEX lexical database. The German and Dutch lexicons of CELEX are larger than the English lexicon. For both languages, we selected a 25K test set, and two different training sets, one containing 50K words and the other containing 250K words. The results are presented in Table 3.

Method	German	Dutch
MAXONSET	19.51	23.44
SONORITY	76.32	77.51
LEGALITY	79.55	64.31
SVM-HMM (50K words)	99.26	97.79
SVM-HMM (250K words)	99.87	99.16

Table 3: Word accuracy on the CELEX dataset.

While our SVM-HMM approach is entirely language independent, the same cannot be said about other methods. The maximal onset principle appears to hold much more strongly for English than for German and Dutch (e.g., *patron*: [pe-trən] vs. [pat-ron]). LEGALITY and SONORITY also appear to be less effective, possibly because of greater tendency for syllabifications to match morphological boundaries (e.g., English *exclusive*: [ɪk-sklu-sɪv] vs. Dutch *exclusief* [ɛks-kly-zif]). SONORITY is further affected by our decision to employ the constraints of Kenstowicz (1994), although they clearly pertain to English. We expect that adapting them to specific languages would bring the results closer to the level of the English experiments.

Although our SVM system is tuned using an English development set, the results on both German and Dutch are excellent. We could not find any quantitative data for comparisons on Dutch, but the comparison with the previously reported results on German CELEX demonstrates the quality of our approach. The numbers that follow refer to 10-fold cross-validation on the entire lexicon (over 320K entries) unless noted otherwise. Krenn (1997) obtains *tag* accuracy of 98.34%, compared to our system's tag accuracy of 99.97% when trained on 250K words. With a hand-crafted grammar, Müller (2002) achieves 96.88% word accuracy on CELEX-derived syllabifications, with a training corpus of two million tokens. Without a hand-crafted grammar, she reports 90.45% word accuracy (Müller, 2006). Applying a standard smoothing algorithm and fourth-order HMM, Demberg (2006) scores 98.47% word accuracy. A fifth-order joint N-gram model of Schmid et al. (2007) achieves 99.85% word accuracy with about 278K training points. However, unlike generative approaches, our

Method	English	German
SONORITY	97.0	94.2
SVM-HMM	99.9	99.4
Categorical Parser	94.9	92.7
Maximum Likelihood	98.1	97.4

Table 4: Word accuracy on the datasets of Goldwater and Johnson (2005).

SVM-HMM can condition each emission on large portions of the input using only a first-order Markov model, which implies much faster syllabification performance.

6.4 Direct Comparison with an MLE approach

The results of the competitive approaches that have been quoted so far (with the exception of *tsylb*) are not directly comparable, because neither the respective implementations, nor the actual train-test splits are publicly available. However, we managed to obtain the English and German data sets used by Goldwater and Johnson (2005) in their study, which focused primarily on unsupervised syllabification. Their experimental framework is similar to (Müller, 2001). They collect words from running text and create a training set of 20K tokens and a test set of 10K tokens. The running text was taken from the Penn WSJ and ECI corpora, and the syllabified phonemic transcriptions were obtained from CELEX. Table 4 compares our experimental results with their reported results obtained with: (a) supervised Maximum Likelihood training procedures, and (b) a Categorical Syllable Parser implementing the principles of sonority sequencing and onset maximization without Kenstowicz's (1994) onset constraints.

The accuracy figures in Table 4 are noticeably higher than in Table 1. This stems from fundamental differences in the experimental set-up; Goldwater and Johnson (2005) test on tokens as found in text, therefore many frequent short words are duplicated. Furthermore, some words occur during both training and testing, to the benefit of the supervised systems (SVM-HMM and Maximum Likelihood). Nevertheless, the results confirm the level of improvement obtained by both our categorical and supervised approaches.

7 Conclusion

We have presented several different approaches to the syllabification of phonemes. The results of our linguistically-motivated algorithms, show that it is possible to achieve adequate syllabification word accuracy in English with no little or no syllable-annotated training data. We have demonstrated that the poor performance of categorical methods on English NETtalk actually points to problems with the NETtalk annotations, rather than with the methods themselves.

We have also shown that SVM-HMMs can be used to great effect when syllabifying phonemes. In addition to being both efficient and language-independent, they establish a new state-of-the-art for English and Dutch syllabification. However, they do require thousands of labeled training examples to achieve this level of accuracy. In the future, we plan to explore a hybrid approach, which would benefit from both the generality of linguistic principles and the smooth exception-handling of supervised techniques, in order to make best use of whatever data is available.

Acknowledgements

We are grateful to Sharon Goldwater for providing the experimental data sets for comparison. This research was supported by the Natural Sciences and Engineering Research Council of Canada and the Alberta Informatics Circle of Research Excellence.

References

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Hidden markov support vector machines. *Proceedings of the 20Th International Conference on Machine Learning (ICML)*.

R. Baayen, R. Piepenbrock, and L. Gulikers. 1995. The CELEX lexical database (CD-ROM).

Susan Bartlett, Grzegorz Kondrak, and Colin Cherry. 2008. Automatic syllabification with structured SVMs for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*, pages 568–576, Columbus, Ohio.

Susan Bartlett. 2007. Discriminative approach to automatic syllabification. Master's thesis, Department of Computing Science, University of Alberta.

Juliette Blevins. 1995. The syllable in phonological theory. In John Goldsmith, editor, *The handbook of phonological theory*, pages 206–244. Blackwell.

Gosse Bouma. 2002. Finite state methods for hyphenation. *Natural Language Engineering*, 1:1–16.

David Crystal. 2003. *A dictionary of linguistics and phonetics*. Blackwell.

Walter Daelemans and Antal van den Bosch. 1992. Generalization performance of backpropagaion learning on a syllabification task. In *Proceedings of the 3rd Twente Workshop on Language Technology*, pages 27–38.

Walter Daelemans, Antal van den Bosch, and Ton Weijters. 1997. IGTree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, pages 407–423.

Robert Damper. 2001. Learning about speech from data: Beyond NETtalk. In *Data-Driven Techniques in Speech Synthesis*, pages 1–25. Kluwer Academic Publishers.

Vera Demberg. 2006. Letter-to-phoneme conversion for a German text-to-speech system. Master's thesis, University of Stuttgart.

William Fisher. 1996. Tsylb syllabification package. ftp://jaguar.ncsl.nist.gov/pub/tsylb2-1.1.tar.Z. Last accessed 31 March 2008.

Sharon Goldwater and Mark Johnson. 2005. Representational bias in usupervised learning of syllable structure. In *Prcoeedings of the 9th Conference on Computational Natural Language Learning (CoNLL)*, pages 112–119.

Jeremy Goslin and Ulrich Frauenfelder. 2001. A comparison of theoretical and human syllabification. *Language and Speech*, 44:409–436.

Carmen Jany, Matthew Gordon, Carlos M Nash, and Nobutaka Takara. 2007. How universal is the sonority hierarchy? A cross-linguistic study. In *16th International Congress of Phonetic Sciences*, pages 1401–1404.

Daniel Kahn. 1976. *Syllable-based generalizations in English Phonology*. Ph.D. thesis, Indiana University.

Michael Kenstowicz. 1994. *Phonology in Generative Grammar*. Blackwell.

George Kiraz and Bernd Möbius. 1998. Multilingual syllabification using weighted finite-state transducers. In *Proceedings of the 3rd Workshop on Speech Synthesis*.

Brigitte Krenn. 1997. Tagging syllables. In *Proceedings of Eurospeech*, pages 991–994.

Yannick Marchand and Robert Damper. 2007. Can syllabification improve pronunciation by analogy of English? *Natural Language Engineering*, 13(1):1–24.

Yannick Marchand, Connie Adsett, and Robert Damper. 2007. Automatic syllabification in English: A comparison of different algorithms. *Language and Speech*. To appear.

Karin Müller, Bernd Möbius, and Detlef Prescher. 2000. Inducing probabilistic syllable classes using multivariate clustering. In *Prcoeedings of the 38th meeting of the ACL*.

Karin Müller. 2001. Automatic detection of syllable boundaries combining the advantages of treebank and bracketed corpora training. *Proceedings on the 39Th Meeting of the ACL*.

Karin Müller. 2002. Probabilistic context-free grammars for phonology. *Proceedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON)*, pages 80–90.

Karin Müller. 2006. Improving syllabification models with phonotactic knowledge. *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology At HLT-NAACL*.

Steve Pearson, Roland Kuhn, Steven Fincke, and Nick Kibre. 2000. Automatic methods for lexical stress assignment and syllabification. In *Proceedings of the 6th International Conference on Spoken Language Processing (ICSLP)*.

Helmut Schmid, Bernd Möbius, and Julia Weidenkaff. 2007. Tagging syllable boundaries with joint N-gram models. In *Proceedings of Interspeech*.

Elisabeth Selkirk. 1984. On the major class features and syllable theory. In *Language Sound Structure*. MIT Press.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. *Proceedings of the 21st International Conference on Machine Learning (ICML)*.

Antal van den Bosch. 1997. *Learning to pronounce written words: a study in inductive language learning*. Ph.D. thesis, Universiteit Maastricht.

Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars

Mark Johnson
Brown University
Providence, RI
Mark_Johnson@Brown.edu

Sharon Goldwater
University of Edinburgh
Edinburgh EH8 9AB
sgwater@inf.ed.ac.uk

Abstract

One of the reasons nonparametric Bayesian inference is attracting attention in computational linguistics is because it provides a principled way of learning the units of generalization together with their probabilities. Adaptor grammars are a framework for defining a variety of hierarchical nonparametric Bayesian models. This paper investigates some of the choices that arise in formulating adaptor grammars and associated inference procedures, and shows that they can have a dramatic impact on performance in an unsupervised word segmentation task. With appropriate adaptor grammars and inference procedures we achieve an 87% word token f-score on the standard Brent version of the Bernstein-Ratner corpus, which is an error reduction of over 35% over the best previously reported results for this corpus.

1 Introduction

Most machine learning algorithms used in computational linguistics are *parametric*, i.e., they learn a numerical weight (e.g., a probability) associated with each feature, where the set of features is fixed before learning begins. Such procedures can be used to learn features or structural units by embedding them in a "propose-and-prune" algorithm: a feature proposal component proposes potentially useful features (e.g., combinations of the currently most useful features), which are then fed to a parametric learner that estimates their weights. After estimating feature weights and pruning "useless" low-weight features, the cycle repeats. While such algorithms can achieve impressive results (Stolcke and Omohundro, 1994), their effectiveness depends on how well the feature proposal step relates to the overall learning objective, and it can take considerable insight and experimentation to devise good feature proposals.

One of the main reasons for the recent interest in *nonparametric Bayesian inference* is that it offers a systematic framework for structural inference, i.e., inferring the features relevant to a particular problem as well as their weights. (Here "nonparametric" means that the models do not have a fixed set of parameters; our nonparametric models do have parameters, but the particular parameters in a model are learned along with their values). Dirichlet Processes and their associated predictive distributions, Chinese Restaurant Processes, are one kind of nonparametric Bayesian model that has received considerable attention recently, in part because they can be composed in hierarchical fashion to form Hierarchical Dirichlet Processes (HDP) (Teh et al., 2006).

Lexical acquisition is an ideal test-bed for exploring methods for inferring structure, where the features learned are the words of the language. (Even the most hard-core nativists agree that the words of a language must be learned). We use the unsupervised word segmentation problem as a test case for evaluating structural inference in this paper. Nonparametric Bayesian methods produce state-of-the-art performance on this task (Goldwater et al., 2006a; Goldwater et al., 2007; Johnson, 2008).

In a computational linguistics setting it is natural to try to align the HDP hierarchy with the hierarchy defined by a grammar. Adaptor grammars, which are one way of doing this, make it easy to explore a wide variety of HDP grammar-based models. Given an appropriate adaptor grammar, the fea-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 317–325,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

tures learned by adaptor grammars can correspond to linguistic units such as words, syllables and collocations. Different adaptor grammars encode different assumptions about the structure of these units and how they relate to each other. A generic adaptor grammar inference program infers these units from training data, making it easy to investigate how these assumptions affect learning (Johnson, 2008).[1]

However, there are a number of choices in the design of adaptor grammars and the associated inference procedure. While this paper studies the impact of these on the word segmentation task, these choices arise in other nonparametric Bayesian inference problems as well, so our results should be useful more generally. The rest of this paper is organized as follows. The next section reviews adaptor grammars and presents three different adaptor grammars for word segmentation that serve as running examples in this paper. Adaptor grammars contain a large number of adjustable parameters, and Section 3 discusses how these can be estimated using Bayesian techniques. Section 4 examines several implementation options within the adaptor grammar inference algorithm and shows that they can make a significant impact on performance. Cumulatively these changes make a significant difference in word segmentation accuracy: our final adaptor grammar performs unsupervised word segmentation with an 87% token f-score on the standard Brent version of the Bernstein-Ratner corpus (Bernstein-Ratner, 1987; Brent and Cartwright, 1996), which is an error reduction of over 35% compared to the best previously reported results on this corpus.

2 Adaptor grammars

This section informally introduces adaptor grammars using unsupervised word segmentation as a motivating application; see Johnson et al. (2007b) for a formal definition of adaptor grammars.

Consider the problem of learning language from continuous speech: segmenting each utterance into words is a nontrivial problem that language learners must solve. Elman (1990) introduced an idealized version of this task, and Brent and Cartwright (1996) presented a version of it where the data consists of unsegmented phonemic representations of the sentences in the Bernstein-Ratner corpus of child-directed speech (Bernstein-Ratner, 1987). Because these phonemic representations are obtained by looking up orthographic forms in a pronouncing dictionary and appending the results, identifying the word tokens is equivalent to finding the locations of the word boundaries. For example, the phoneme string corresponding to "you want to see the book" (with its correct segmentation indicated) is as follows:

$$\mathrm{y}\ \vartriangle\ \mathrm{u}\ \blacktriangle\ \mathrm{w}\ \vartriangle\ \mathrm{a}\ \vartriangle\ \mathrm{n}\ \vartriangle\ \mathrm{t}\ \blacktriangle\ \mathrm{t}\ \vartriangle\ \mathrm{u}\ \blacktriangle\ \mathrm{s}\ \vartriangle\ \mathrm{i}\ \blacktriangle\ \mathrm{D}\ \vartriangle\ 6\ \blacktriangle\ \mathrm{b}\ \vartriangle\ \mathrm{U}\ \vartriangle\ \mathrm{k}$$

We can represent any possible segmentation of any possible sentence as a tree generated by the following *unigram grammar*.

$$\begin{aligned}\mathrm{Sentence} &\rightarrow \underline{\mathrm{Word}}^{+}\\ \underline{\mathrm{Word}} &\rightarrow \mathrm{Phoneme}^{+}\end{aligned}$$

The nonterminal Phoneme expands to each possible phoneme; the underlining, which identifies "adapted nonterminals", will be explained below. In this paper "+" abbreviates right-recursion through a dummy nonterminal, i.e., the unigram grammar actually is:

$$\begin{aligned}\mathrm{Sentence} &\rightarrow \underline{\mathrm{Word}}\\ \mathrm{Sentence} &\rightarrow \underline{\mathrm{Word}}\ \mathrm{Sentence}\\ \underline{\mathrm{Word}} &\rightarrow \mathrm{Phonemes}\\ \mathrm{Phonemes} &\rightarrow \mathrm{Phoneme}\\ \mathrm{Phonemes} &\rightarrow \mathrm{Phoneme}\ \mathrm{Phonemes}\end{aligned}$$

A PCFG with these productions can represent all possible segmentations of any Sentence into a sequence of Words. But because it assumes that the probability of a word is determined purely by multiplying together the probability of its individual phonemes, it has no way to encode the fact that certain strings of phonemes (the words of the language) have much higher probabilities than other strings containing the same phonemes. In order to do this, a PCFG would need productions like the following one, which encodes the fact that "want" is a Word.

$$\mathrm{Word} \rightarrow \mathrm{w}\ \mathrm{a}\ \mathrm{n}\ \mathrm{t}$$

Adaptor grammars can be viewed as a way of formalizing this idea. Adaptor grammars learn the probabilities of entire subtrees, much as in tree substitution grammar (Joshi, 2003) and DOP (Bod,

[1]The adaptor grammar inference program is available for download at http://www.cog.brown.edu/~mj/Software.htm.

1998). (For computational efficiency reasons adaptor grammars require these subtrees to expand to terminals). The set of possible adapted tree fragments is the set of all subtrees generated by the CFG whose root label is a member of the set of *adapted nonterminals* A (adapted nonterminals are indicated by underlining in this paper). For example, in the unigram adaptor grammar $A = \{\mathrm{Word}\}$, which means that the adaptor grammar inference procedure learns the probability of each possible Word subtree. Thus adaptor grammars are simple models of structure learning in which adapted subtrees are the units of generalization.

One might try to reduce adaptor grammar inference to PCFG parameter estimation by introducing a context-free rule for each possible adapted subtree, but such an attempt would fail because the number of such adapted subtrees, and hence the number of corresponding rules, is unbounded. However nonparametric Bayesian inference techniques permit us to *sample* from this infinite set of adapted subtrees, and only require us to instantiate the finite number of them needed to analyse the finite training data.

An *adaptor grammar* is a 7-tuple $(N, W, R, S, \boldsymbol{\theta}, A, \boldsymbol{C})$ where $(N, W, R, S, \boldsymbol{\theta})$ is a PCFG with nonterminals N, terminals W, rules R, start symbol $S \in N$ and rule probabilities $\boldsymbol{\theta}$, where θ_r is the probability of rule $r \in R$, $A \subseteq N$ is the set of *adapted nonterminals* and $\boldsymbol{C}$ is a vector of *adaptors* indexed by elements of A, so C_X is the adaptor for adapted nonterminal $X \in A$.

Informally, an adaptor C_X nondeterministically maps a stream of trees from a *base distribution* H_X whose support is $\mathcal{T}_X$ (the set of subtrees whose root node is $X \in N$ generated by the grammar's rules) into another stream of trees whose support is also $\mathcal{T}_X$. In adaptor grammars the base distributions H_X are determined by the PCFG rules expanding X and the other adapted distributions, as explained in Johnson et al. (2007b). When called upon to generate another sample tree, the adaptor either generates and returns a fresh tree from H_X or regenerates a tree it has previously emitted, so in general the adapted distribution differs from the base distribution.

This paper uses adaptors based on Chinese Restaurant Processes (CRPs) or Pitman-Yor Processes (PYPs) (Pitman, 1995; Pitman and Yor, 1997; Ishwaran and James, 2003). CRPs and PYPs nondeterministically generate infinite sequences of natural numbers $z_1, z_2, \ldots$, where $z_1 = 1$ and each $z_{n+1} \leq m + 1$ where $m = \max(z_1, \ldots, z_n)$. In the "Chinese Restaurant" metaphor samples produced by the adaptor are viewed as "customers" and z_n is the index of the "table" that the nth customer is seated at. In adaptor grammars each table in the adaptor C_X is labeled with a tree sampled from the base distribution H_X that is shared by all customers at that table; thus the nth sample tree from the adaptor C_X is the z_nth sample from H_X.

CRPs and PYPs differ in exactly how the sequence $\{z_k\}$ is generated. Suppose $\boldsymbol{z} = (z_1, \ldots, z_n)$ have already been generated and $m = \max(\boldsymbol{z})$. Then a CRP generates the next table index z_{n+1} according to the following distribution:

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{z}) \propto \begin{cases} n_k(\boldsymbol{z}) & \text{if } k \leq m \\ \alpha & \text{if } k = m + 1 \end{cases}$$

where $n_k(\boldsymbol{z})$ is the number of times table k appears in $\boldsymbol{z}$ and $\alpha > 0$ is an adjustable parameter that determines how often a new table is chosen. This means that if C_X is a CRP adaptor then the next tree t_{n+1} it generates is the same as a previously generated tree t' with probability proportional to the number of times C_X has generated t' before, and is a "fresh" tree t sampled from H_X with probability proportional to $\alpha_X H_X(t)$. This leads to a powerful "rich-get-richer" effect in which popular trees are generated with increasingly high probabilities.

Pitman-Yor Processes can control the strength of this effect somewhat by moving mass from existing tables to the base distribution. The PYP predictive distribution is:

$$\mathrm{P}(Z_{n+1} = k \mid \boldsymbol{z}) \propto \begin{cases} n_k(\boldsymbol{z}) - a & \text{if } k \leq m \\ m\,a + b & \text{if } k = m + 1 \end{cases}$$

where $a \in [0, 1]$ and $b > 0$ are adjustable parameters. It's easy to see that the CRP is a special case of the PRP where $a = 0$ and $b = \alpha$.

Each adaptor in an adaptor grammar can be viewed as estimating the probability of each adapted subtree t; this probability can differ substantially from t's probability $H_X(t)$ under the base distribution. Because Words are adapted in the unigram adaptor grammar it effectively estimates the probability of each Word tree separately; the sampling estimators described in section 4 only instantiate those Words actually used in the analysis of $\mathrm{Sentences}$ in the corpus. While the Word adaptor will generally

prefer to reuse Words that have been used elsewhere in the corpus, it is always possible to generate a fresh Word using the CFG rules expanding Word into a string of Phonemes.

We assume for now that all CFG rules R_X expanding the nonterminal $X \in N$ have the same probability (although we will explore estimating $\boldsymbol{\theta}$ below), so the base distribution H_{Word} is a "monkeys banging on typewriters" model. That means the unigram adaptor grammar implements the Goldwater et al. (2006a) unigram word segmentation model, and in fact it produces segmentations of similar accuracies, and exhibits the same characteristic undersegmentation errors. As Goldwater et al. point out, because Words are the only units of generalization available to a unigram model it tends to misanalyse collocations as words, resulting in a marked tendancy to undersegment.

Goldwater et al. demonstrate that modelling bigram dependencies mitigates this undersegmentation. While adaptor grammars cannot express the Goldwater et al. bigram model, they can get much the same effect by directly modelling collocations (Johnson, 2008). A *collocation adaptor grammar* generates a Sentence as a sequence of Collocations, each of which expands to a sequence of Words.

$$\begin{aligned}\text{Sentence} &\rightarrow \underline{\text{Colloc}}^+\\ \underline{\text{Colloc}} &\rightarrow \underline{\text{Word}}^+\\ \underline{\text{Word}} &\rightarrow \text{Phoneme}^+\end{aligned}$$

Because Colloc is adapted, the collocation adaptor grammar learns Collocations as well as Words. (Presumably these approximate syntactic, semantic and pragmatic interword dependencies). Johnson reported that the collocation adaptor grammar segments as well as the Goldwater et al. bigram model, which we confirm here.

Recently other researchers have emphasised the utility of phonotactic constraints (i.e., modeling the allowable phoneme sequences at word onsets and endings) for word segmentation (Blanchard and Heinz, 2008; Fleck, 2008). Johnson (2008) points out that adaptor grammars that model words as sequences of syllables can learn and exploit these constraints, significantly improving segmentation accuracy. Here we present an adaptor grammar that models collocations together with these phonotactic constraints. This grammar is quite complex, permitting us to study the effects of the various model and implementation choices described below on a complex hierarchical nonparametric Bayesian model.

The *collocation-syllable adaptor grammar* generates a Sentence in terms of three levels of Collocations (enabling it to capture a wider range of interword dependencies), and generates Words as sequences of 1 to 4 Syllables. Syllables are subcategorized as to whether they are initial (I), final (F) or both (IF).

$$\begin{aligned}\text{Sentence} &\rightarrow \underline{\text{Colloc3}}^+\\ \underline{\text{Colloc3}} &\rightarrow \underline{\text{Colloc2}}^+\\ \underline{\text{Colloc2}} &\rightarrow \underline{\text{Colloc1}}^+\\ \underline{\text{Colloc1}} &\rightarrow \underline{\text{Word}}^+\\ \underline{\text{Word}} &\rightarrow \text{SyllableIF}\\ \underline{\text{Word}} &\rightarrow \text{SyllableI (Syllable) (Syllable) SyllableF}\\ \text{Syllable} &\rightarrow \underline{\text{Onset}}\ \text{Rhyme}\\ \underline{\text{Onset}} &\rightarrow \text{Consonant}^+\\ \text{Rhyme} &\rightarrow \underline{\text{Nucleus}}\ \underline{\text{Coda}}\\ \underline{\text{Nucleus}} &\rightarrow \text{Vowel}^+\\ \underline{\text{Coda}} &\rightarrow \text{Consonant}^+\\ \text{SyllableIF} &\rightarrow \underline{\text{OnsetI}}\ \text{RhymeF}\\ \underline{\text{OnsetI}} &\rightarrow \text{Consonant}^+\\ \text{RhymeF} &\rightarrow \underline{\text{Nucleus}}\ \underline{\text{CodaF}}\\ \underline{\text{CodaF}} &\rightarrow \text{Consonant}^+\\ \text{SyllableI} &\rightarrow \underline{\text{OnsetI}}\ \text{Rhyme}\\ \text{SyllableF} &\rightarrow \underline{\text{Onset}}\ \text{RhymeF}\end{aligned}$$

Here Consonant and Vowel expand to all possible consonants and vowels respectively, and the parentheses in the expansion of Word indicate optionality. Because Onsets and Codas are adapted, the collocation-syllable adaptor grammar learns the possible consonant sequences that begin and end syllables. Moreover, because Onsets and Codas are subcategorized based on whether they are word-peripheral, the adaptor grammar learns which consonant clusters typically appear at word boundaries, even though the input contains no explicit word boundary information (apart from what it can glean from the sentence boundaries).

3 Bayesian estimation of adaptor grammar parameters

Adaptor grammars as defined in section 2 have a large number of free parameters that have to be chosen by the grammar designer; a rule probability θ_r for each PCFG rule $r \in R$ and either one or two hyperparameters for each adapted nonterminal $X \in A$, depending on whether Chinese Restaurant

or Pitman-Yor Processes are used as adaptors. It's difficult to have intuitions about the appropriate settings for the latter parameters, and finding the optimal values for these parameters by some kind of exhaustive search is usually computationally impractical. Previous work has adopted an expedient such as parameter tying. For example, Johnson (2008) set $\boldsymbol{\theta}$ by requiring all productions expanding the same nonterminal to have the same probability, and used Chinese Restaurant Process adaptors with tied parameters α_X, which was set using a grid search.

We now describe two methods of dealing with the large number of parameters in these models that are both more principled and more practical than the approaches described above. First, we can integrate out $\boldsymbol{\theta}$, and second, we can infer values for the adaptor hyperparameters using sampling. These methods (the latter in particular) make it practical to use Pitman-Yor Process adaptors in complex grammars such as the collocation-syllable adaptor grammar, where it is impractical to try to find optimal parameter values by grid search. As we will show, they also improve segmentation accuracy, sometimes dramatically.

3.1 Integrating out θ

Johnson et al. (2007a) describe Gibbs samplers for Bayesian inference of PCFG rule probabilities $\boldsymbol{\theta}$, and these techniques can be used directly with adaptor grammars as well. Just as in that paper, we place Dirichlet priors on $\boldsymbol{\theta}$: here $\boldsymbol{\theta}_X$ is the subvector of $\boldsymbol{\theta}$ corresponding to rules expanding nonterminal $X \in N$, and $\boldsymbol{\beta}_X$ is a corresponding vector of positive real numbers specifying the hyperparameters of the corresponding Dirichlet distributions:

$$\mathrm{P}(\boldsymbol{\theta} \mid \boldsymbol{\beta}) \quad = \quad \prod_{X \in N} \mathrm{Dir}(\boldsymbol{\theta}_X \mid \boldsymbol{\beta}_X)$$

Because the Dirichlet distribution is conjugate to the multinomial distribution, it is possible to integrate out the rule probabilities $\boldsymbol{\theta}$, producing the "collapsed sampler" described in Johnson et al. (2007a).

In our experiments we chose an uniform prior $\beta_r = 1$ for all rules $r \in R$. As Table 1 shows, integrating out $\boldsymbol{\theta}$ only has a major effect on results when the adaptor hyperparameters themselves are not sampled, and even then it did not have a large effect on the collocation-syllable adaptor grammar. This is not too surprising: because the Onset, $\mathrm{Nucleus}$ and Coda adaptors in this grammar learn the probabilities of these building blocks of words, the phoneme probabilities (which is most of what $\boldsymbol{\theta}$ encodes) play less important a role.

3.2 Slice sampling adaptor hyperparameters

As far as we know, there are no conjugate priors for the adaptor hyperparameters a_X or b_X (which corresponds to α_X in a Chinese Restaurant Process), so it is not possible to integrate them out as we did with the rule probabilities $\boldsymbol{\theta}$. However, it is possible to perform Bayesian inference by putting a prior on them and sampling their values.

Because we have no strong intuitions about the values of these parameters we chose uninformative priors. We chose a uniform $\mathrm{Beta}(1, 1)$ prior on a_X, and a "vague" $\mathrm{Gamma}(10, 0.1)$ prior on $b_X = \alpha_X$ (MacKay, 2003). (We experimented with other parameters in the Gamma prior, but found no significant difference in performance).

After each Gibbs sweep through the parse trees $\boldsymbol{t}$ we resampled each of the adaptor parameters from the posterior distribution of the parameter using a slice sampler 10 times. For example, we resample each b_X from:

$$\mathrm{P}(b_X \mid \boldsymbol{t}) \quad \propto \quad \mathrm{P}(\boldsymbol{t} \mid b_X)\, \mathrm{Gamma}(b_X \mid 10, 0.1)$$

Here $\mathrm{P}(\boldsymbol{t} \mid b_X)$ is the likelihood of the current sequence of sample parse trees (we only need the factors that depend on b_X) and $\mathrm{Gamma}(b_X \mid 10, 0.1)$ is the prior. The same formula is used for sampling a_X, except that the prior is now a flat $\mathrm{Beta}(1, 1)$ distribution.

In general we cannot even compute the normalizing constants for these posterior distributions, so we chose a sampler that does not require this. We use a slice sampler here because it does not require a proposal distribution (Neal, 2003). (We initially tried a Metropolis-Hastings sampler but were unable to find a proposal distribution that had reasonable acceptance ratios for all of our adaptor grammars).

As Table 1 makes clear, sampling the adaptor parameters makes a significant difference, especially on the collocation-syllable adaptor grammar. This is not surprising, as the adaptors in that grammar play many different roles and there is no reason to to expect the optimal values of their parameters to be similar.

Condition					Word token f-scores					
					Sample average			Max. Marginal		
Batch initialization	Table label resampling	Integrate out $\boldsymbol{\theta}$	Sample $\alpha_X = b_X$	Sample a_X	unigram	colloc	colloc-syll	unigram	colloc	colloc-syll
•	•	•	•	•	0.55	0.74	0.85	0.56	0.76	0.87
•	•	•	•		0.55	0.72	0.84	0.56	0.74	0.84
•	•	•			0.55	0.72	0.78	0.57	0.75	0.78
•	•				0.54	0.66	0.75	0.56	0.69	0.76
•	•		•	•	0.54	0.70	0.87	0.56	0.74	0.88
•		•	•	•	0.55	0.42	0.54	0.57	0.51	0.55
	•	•	•	•	0.74	0.83	0.88	0.81	0.86	0.89
		•	•	•	0.75	0.43	0.74	0.80	0.56	0.82
			•	•	0.71	0.41	0.76	0.77	0.49	0.82
	•		•	•	0.71	0.73	0.87	0.77	0.75	0.88

Table 1: Word segmentation accuracy measured by word token f-scores on Brent's version of the Bernstein-Ratner corpus as a function of adaptor grammar, adaptor and estimation procedure. Pitman-Yor Process adaptors were used when a_X was sampled, otherwise Chinese Restaurant Process adaptors were used. In runs where $\boldsymbol{\theta}$ was not integrated out it was set uniformly, and all $\alpha_X = b_X$ were set to 100 they were not sampled.

4 Inference for adaptor grammars

Johnson et al. (2007b) describe the basic adaptor grammar inference procedure that we use here. That paper leaves unspecified a number of implementation details, which we show can make a crucial difference to segmentation accuracy. The adaptor grammar algorithm is basically a Gibbs sampler of the kind widely used for nonparametric Bayesian inference (Blei et al., 2004; Goldwater et al., 2006b; Goldwater et al., 2006a), so it seems reasonable to expect that at least some of the details discussed below will be relevant to other applications as well.

The inference algorithm maintains a vector $\boldsymbol{t} = (t_1, \ldots, t_n)$ of sample parses, where $t_i \in \mathcal{T}_S$ is a parse for the ith sentence w_i. It repeatedly chooses a sentence w_i at random and resamples the parse tree t_i for w_i from $\mathrm{P}(t_i \mid \boldsymbol{t}_{-i}, w_i)$, i.e., conditioned on w_i and the parses $\boldsymbol{t}_{-i}$ of all sentences *except* w_i.

4.1 Maximum marginal decoding

Sampling algorithms like ours produce a stream of samples from the posterior distribution over parses of the training data. It is standard to take the output of the algorithm to be the last sample produced, and evaluate those parses. In some other applications of nonparametric Bayesian inference involving latent structure (e.g., clustering) it is difficult to usefully exploit multiple samples, but that is not the case here.

In *maximum marginal decoding* we map each sample parse tree t onto its corresponding word segmentation s, marginalizing out irrelevant detail in t. (For example, the collocation-syllable adaptor grammar contains a syllabification and collocational structure that is irrelevant for word segmentation). Given a set of sample parse trees for a sentence we compute the set of corresponding word segmentations, and return the one that occurs most frequently (this is a sampling approximation to the maximum probability marginal structure).

For each setting in the experiments described in Table 1 we ran 8 samplers for 2,000 iterations (i.e., passes through the training data), and kept the sample parse trees from every 10th iteration after iteration 1000, resulting in 800 sample parses for every sentence. (An examination of the posterior probabilities suggests that all of the samplers using batch initialization and table label resampling had "burnt

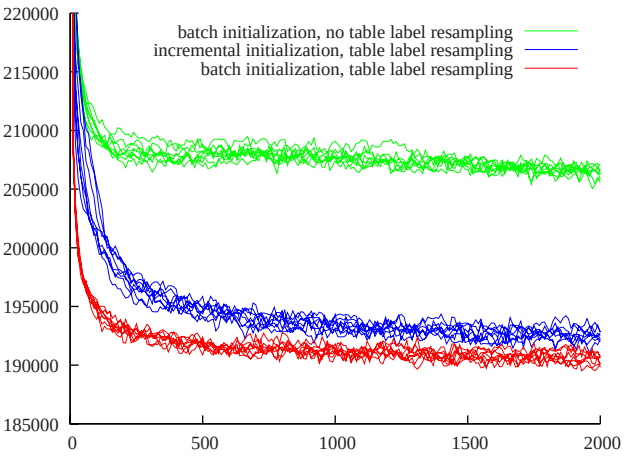


Figure 1: Negative log posterior probability (lower is better) as a function of iteration for 24 runs of the collocation adaptor grammar samplers with Pitman-Yor adaptors. The upper 8 runs use batch initialization but no table label resampling, the middle 8 runs use incremental initialization and table label resampling, while the lower 8 runs use batch initialization and table label resampling.

in" by iteration 1000). We evaluated the word token f-score of the most frequent marginal word segmentation, and compared that to average of the word token f-score for the 800 samples, which is also reported in Table 1. For each grammar and setting we tried, the maximum marginal segmentation was better than the sample average, sometimes by a large margin. Given its simplicity, this suggests that maximum marginal decoding is probably worth trying when applicable.

4.2 Batch initialization

The Gibbs sampling algorithm is initialized with a set of sample parses $\boldsymbol{t}$ for each sentence in the training data. While the fundamental theorem of Markov Chain Monte Carlo guarantees that eventually samples will converge to the posterior distribution, it says nothing about how long the "burn in" phase might last (Robert and Casella, 2004). In practice initialization can make a huge difference to the performance of Gibbs samplers (just as it can with other unsupervised estimation procedures such as Expectation Maximization).

There are many different ways in which we could generate the initial trees $\boldsymbol{t}$; we only study two of the obvious methods here. *Batch initialization* assigns every sentence a random parse tree in parallel. In more detail, the initial parse tree t_i for sentence w_i is sampled from $\mathrm{P}(t \mid w_i, G')$, where G' is the PCFG obtained from the adaptor grammar by ignoring its last two components A and C (i.e., the adapted nonterminals and their adaptors), and seated at a new table. This means that in batch initialization each initial parse tree is randomly generated without any adaptation at all.

Incremental initialization assigns the initial parse trees t_i to sentences w_i in order, updating the adaptor grammar as it goes. That is, t_i is sampled from $\mathrm{P}(t \mid w_i, t_1, \ldots, t_{i-1})$. This is easy to do in the context of Gibbs sampling, since this distribution is a minor variant of the distribution $\mathrm{P}(t_i \mid \boldsymbol{t}_{-i}, w_i)$ used during Gibbs sampling itself.

Incremental initialization is greedier than batch initialization, and produces initial sample trees with much higher probability. As Table 1 shows, across all grammars and conditions after 2,000 iterations incremental initialization produces samples with much better word segmentation token f-score than does batch initialization, with the largest improvement on the unigram adaptor grammar.

However, incremental initialization results in sample parses with lower posterior probability for the unigram and collocation adaptor grammars (but not for the collocation-syllable adaptor grammar). Figure 1 plots the posterior probabilities of the sample trees $\boldsymbol{t}$ at each iteration for the collocation adaptor grammar, showing that even after 2,000 iterations incremental initialization results in trees that are much less likely than those produced by batch initialization. It seems that with incremental initialization the Gibbs sampler gets stuck in a local optimum which it is extremely unlikely to move away from.

It is interesting that incremental initialization results in more accurate word segmentation, even though the trees it produces have lower posterior probability. This seems to be because the most probable analyses produced by the unigram and, to a lesser extent, the collocation adaptor grammars tend to undersegment. Incremental initialization greedily searches for common substrings, and because such substrings are more likely to be short rather than long, it tends to produce analyses with shorter words than batch initialization does. Goldwater et al. (2006a) show that Brent's incremental segmentation algorithm (Brent, 1999) has a similar property.

We favor batch initialization because we are in-

terested in understanding the properties of our models (expressed here as adaptor grammars), and batch initialization does a better job of finding the most probable analyses under these models. However, it might be possible to justify incremental initialization as (say) cognitively more plausible.

4.3 Table label resampling

Unlike the previous two implementation choices which apply to a broad range of algorithms, table label resampling is a specialized kind of Gibbs step for adaptor grammars and similar hierarchical models that is designed to improve mobility. The adaptor grammar algorithm described in Johnson et al. (2007b) repeatedly resamples parses for the *sentences* of the training data. However, the adaptor grammar sampler itself maintains of a hierarchy of Chinese Restaurant Processes or Pitman-Yor Processes, one per adapted nonterminal $X \in A$, that cache subtrees from $\mathcal{T}_X$. In general each of these subtrees will occur many times in the parses for the training data sentences. Table label resampling resamples the trees in these adaptors (i.e., the table labels, to use the restaurant metaphor), potentially changing the analysis of many sentences at once. For example, each $\mathrm{Collocation}$ in the collocation adaptor grammar can occur in many $\mathrm{Sentences}$, and each Word can occur in many $\mathrm{Collocations}$. Resampling a single $\mathrm{Collocation}$ can change the way it is analysed into Words, thus changing the analysis of all of the $\mathrm{Sentences}$ containing that $\mathrm{Collocation}$.

Table label resampling is an additional resampling step performed after each Gibbs sweep through the training data in which we resample the parse trees labeling the tables in the adaptor for each $X \in A$. Specifically, if the adaptor C_X for $X \in A$ currently contains m tables labeled with the trees $\boldsymbol{t} = (t_1, \ldots, t_m)$ then table label resampling replaces each $t_j, j \in 1, \ldots, m$ in turn with a tree sampled from $\mathrm{P}(t \mid \boldsymbol{t}_{-j}, w_j)$, where w_j is the terminal yield of t_j. (Within each adaptor we actually resample all of the trees $\boldsymbol{t}$ in a randomly chosen order).

Table label resampling is a kind of Gibbs sweep, but at a higher level in the Bayesian hierarchy than the standard Gibbs sweep. It's easy to show that table label resampling preserves detailed balance for the adaptor grammars presented in this paper, so interposing table label resampling steps with the standard Gibbs steps also preserves detailed balance.

We expect table label resampling to have the greatest impact on models with a rich hierarchical structure, and the experimental results in Table 1 confirm this. The unigram adaptor grammar does not involve nested adapted nonterminals, so we would not expect table label resampling to have any effect on its analyses. On the other hand, the collocation-syllable adaptor grammar involves a rich hierarchical structure, and in fact without table label resampling our sampler did not burn in or mix within 2,000 iterations. As Figure 1 shows, table label resampling produces parses with higher posterior probability, and Table 1 shows that table label resampling makes a significant difference in the word segmentation f-score of the collocation and collocation-syllable adaptor grammars.

5 Conclusion

This paper has examined adaptor grammar inference procedures and their effect on the word segmentation problem. Some of the techniques investigated here, such as batch versus incremental initialization, are quite general and may be applicable to a wide range of other algorithms, but some of the other techniques, such as table label resampling, are specialized to nonparametric hierarchical Bayesian inference. We've shown that sampling adaptor hyperparameters is feasible, and demonstrated that this improves word segmentation accuracy of the collocation-syllable adaptor grammar by almost 10%, corresponding to an error reduction of over 35% compared to the best results presented in Johnson (2008). We also described and investigated table label resampling, which dramatically improves the effectiveness of Gibbs sampling estimators for complex adaptor grammars, and makes it possible to work with adaptor grammars with complex hierarchical structure.

Acknowledgments

We thank Erik Sudderth for suggesting sampling the Pitman-Yor hyperparameters and the ACL reviewers for their insightful comments. This research was funded by NSF awards 0544127 and 0631667 to Mark Johnson.

References

N. Bernstein-Ratner. 1987. The phonology of parent-child speech. In K. Nelson and A. van Kleeck, editors, *Children's Language*, volume 6. Erlbaum, Hillsdale, NJ.

Daniel Blanchard and Jeffrey Heinz. 2008. Improving word segmentation by simultaneously learning phonotactics. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 65–72, Manchester, England, August.

David Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Rens Bod. 1998. *Beyond grammar: an experience-based theory of language*. CSLI Publications, Stanford, California.

M. Brent and T. Cartwright. 1996. Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 61:93–125.

M. Brent. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

Jeffrey Elman. 1990. Finding structure in time. *Cognitive Science*, 14:197–211.

Margaret M. Fleck. 2008. Lexicalized phonotactic word segmentation. In *Proceedings of ACL-08: HLT*, pages 130–138, Columbus, Ohio, June. Association for Computational Linguistics.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006a. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia. Association for Computational Linguistics.

Sharon Goldwater, Tom Griffiths, and Mark Johnson. 2006b. Interpolating between types and tokens by estimating power-law generators. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 459–466, Cambridge, MA. MIT Press.

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2007. Distributional cues to word boundaries: Context is important. In David Bamman, Tatiana Magnitskaia, and Colleen Zaller, editors, *Proceedings of the 31st Annual Boston University Conference on Language Development*, pages 239–250, Somerville, MA. Cascadilla Press.

H. Ishwaran and L. F. James. 2003. Generalized weighted Chinese restaurant processes for species sampling mixture models. *Statistica Sinica*, 13:1211–1235.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007a. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York, April. Association for Computational Linguistics.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007b. Adaptor Grammars: A framework for specifying compositional nonparametric Bayesian models. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648. MIT Press, Cambridge, MA.

Mark Johnson. 2008. Using adaptor grammars to identifying synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, Columbus, Ohio. Association for Computational Linguistics.

Aravind Joshi. 2003. Tree adjoining grammars. In Ruslan Mikkov, editor, *The Oxford Handbook of Computational Linguistics*, pages 483–501. Oxford University Press, Oxford, England.

David J.C. MacKay. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Radford M. Neal. 2003. Slice sampling. *Annals of Statistics*, 31:705–767.

J. Pitman and M. Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900.

J. Pitman. 1995. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102:145–158.

Christian P. Robert and George Casella. 2004. *Monte Carlo Statistical Methods*. Springer.

Andreas Stolcke and Stephen Omohundro. 1994. Inducing probabilistic grammars by Bayesian model merging. In Rafael C. Carrasco and Jose Oncina, editors, *Grammatical Inference and Applications*, pages 106–118. Springer, New York.

Y. W. Teh, M. Jordan, M. Beal, and D. Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581.

Joint Parsing and Named Entity Recognition

Jenny Rose Finkel and Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305
{jrfinkel|manning}@cs.stanford.edu

Abstract

For many language technology applications, such as question answering, the overall system runs several independent processors over the data (such as a named entity recognizer, a coreference system, and a parser). This easily results in inconsistent annotations, which are harmful to the performance of the aggregate system. We begin to address this problem with a joint model of parsing and named entity recognition, based on a discriminative feature-based constituency parser. Our model produces a consistent output, where the named entity spans do not conflict with the phrasal spans of the parse tree. The joint representation also allows the information from each type of annotation to improve performance on the other, and, in experiments with the OntoNotes corpus, we found improvements of up to 1.36% absolute F1 for parsing, and up to 9.0% F1 for named entity recognition.

1 Introduction

In order to build high quality systems for complex NLP tasks, such as question answering and textual entailment, it is essential to first have high quality systems for lower level tasks. A good (deep analysis) question answering system requires the data to first be annotated with several types of information: parse trees, named entities, word sense disambiguation, etc. However, having high performing, low-level systems is not enough; the assertions of the various levels of annotation must be *consistent* with one another. When a named entity span has crossing brackets with the spans in the parse tree it is usually impossible to effectively combine these pieces of information, and system performance suffers. But, unfortunately, it is still common practice to cobble together independent systems for the various types of annotation, and there is no guarantee that their outputs will be consistent.

This paper begins to address this problem by building a joint model of both parsing and named entity recognition. Vapnik has observed (Vapnik, 1998; Ng and Jordan, 2002) that "one should solve the problem directly and never solve a more general problem as an intermediate step," implying that building a joint model of two phenomena is more likely to harm performance on the individual tasks than to help it. Indeed, it has proven very difficult to build a joint model of parsing and semantic role labeling, either with PCFG trees (Sutton and McCallum, 2005) or with dependency trees. The CoNLL 2008 shared task (Surdeanu et al., 2008) was intended to be about joint dependency parsing and semantic role labeling, but the top performing systems decoupled the tasks and outperformed the systems which attempted to learn them jointly. Despite these earlier results, we found that combining parsing and named entity recognition modestly improved performance on both tasks. Our joint model produces an output which has consistent parse structure and named entity spans, and does a better job at both tasks than separate models with the same features.

We first present the joint, discriminative model that we use, which is a feature-based CRF-CFG parser operating over tree structures augmented with NER information. We then discuss in detail how we make use of the recently developed OntoNotes corpus both for training and testing the model, and then finally present the performance of the model and some discussion of what causes its superior performance, and how the model relates to prior work.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 326–334,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

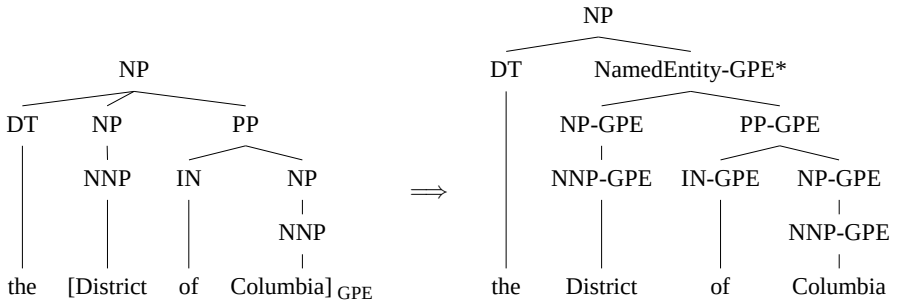


Figure 1: An example of a (sub)tree which is modified for input to our learning algorithm. Starting from the normalized tree discussed in section 4.1, a new *NamedEntity* node is added, so that the named entity corresponds to a single phrasal node. That node, and its descendents, have their labels augmented with the type of named entity. The * on the *NamedEntity* node indicates that it is the root of the named entity.

2 The Joint Model

When constructing a joint model of parsing and named entity recognition, it makes sense to think about how the two distinct levels of annotation may help one another. Ideally, a named entity should correspond to a phrase in the constituency tree. However, parse trees will occasionally lack some explicit structure, such as with right branching NPs. In these cases, a named entity may correspond to a contiguous set of children within a subtree of the entire parse. The one thing that should never happen is for a named entity span to have crossing brackets with any spans in the parse tree.

For named entities, the joint model should help with boundaries. The internal structure of the named entity, and the structural context in which it appears, can also help with determining the type of entity. Finding the best parse for a sentence can be helped by the named entity information in similar ways. Because named entities *should* correspond to phrases, information about them should lead to better bracketing. Also, knowing that a phrase is a named entity, and the type of entity, may help in getting the structural context, and internal structure, of that entity correct.

2.1 Joint Representation

After modifying the OntoNotes dataset to ensure consistency, which we will discuss in Section 4, we augment the parse tree with named entity information, for input to our learning algorithm. In the cases where a named entity corresponds to multiple contiguous children of a subtree, we add a new *NamedEntity* node, which is the new parent to those children. Now, all named entities correspond to a single phrasal node in the entire tree. We then augment the labels of the phrasal node and its descendents with the type of named entity. We also distinguish between the root node of an entity, and the descendent nodes. See Figure 1 for an illustration. This representation has several benefits, outlined below.

2.1.1 Nested Entities

The OntoNotes data does not contain any nested entities. Consider the named entity portions of the rules seen in the training data. These will look, for instance, like *none* → *none person*, and *organization* → *organization organization*. Because we only allow named entity derivations which we have seen in the data, nested entities are impossible. However, there is clear benefit in a representation allowing nested entities. For example, it would be beneficial to recognize that the *United States Supreme Court* is a an *organization*, but that it also contains a nested *GPE*.[1] Fortunately, if we encounter data which has been annotated with nested entities, this representation will be able to handle them in a natural way. In the given example, we would have a derivation which includes *organization* → *GPE organization*. This information will be helpful for correctly labeling nested entities such as *New Jersey Supreme Court*, because the model will learn how nested entities tend to decompose.

2.1.2 Feature Representation for Named Entities

Currently, named entity recognizers are usually constructed using sequence models, with linear chain

[1] As far as we know, GENIA (Kim et al., 2003) is the only corpus currently annotated with nested entities.

conditional random fields (CRFs) being the most common. While it is possible for CRFs to have links that are longer distance than just between adjacent words, most of the benefit is from local features, over the words and labels themselves, and from features over adjacent pairs of words and labels. Our joint representation allows us to port both types of features from such a named entity recognizer. The local features can be computed at the same time the features over parts of speech are computed. These are the leaves of the tree, when only the named entity for the current word is known.[2] The pairwise features, over adjacent labels, are computed at the same time as features over binary rules. Binarization of the tree is necessary for efficient computation, so the trees consist solely of unary and binary productions. Because of this, for all pairs of adjacent words within an entity, there will be a binary rule applied where one word will be under the left child and the other word will be under the right child. Therefore, we compute features over adjacent words/labels when computing the features for the binary rule which joins them.

2.2 Learning the Joint Model

We construct our joint model as an extension to the discriminatively trained, feature-rich, conditional random field-based, CRF-CFG parser of (Finkel and Manning, 2008). Their parser is similar to a chart-based PCFG parser, except that instead of putting probabilities over rules, it puts *clique potentials* over local subtrees. These unnormalized potentials know what span (and split) the rule is over, and arbitrary features can be defined over the local subtree, the span/split and the words of the sentence. The inside-outside algorithm is run over the clique potentials to produce the partial derivatives and normalizing constant which are necessary for optimizing the log likelihood.

2.3 Grammar Smoothing

Because of the addition of named entity annotations to grammar rules, if we use the grammar as read off the treebank, we will encounter problems with sparseness which severely degrade performance. This degradation occurs because of CFG rules which only occur in the training data augmented with named entity information, and because of rules which only occur without the named entity information. To combat this problem, we added extra rules, unseen in the training data.

2.3.1 Augmenting the Grammar

For every rule encountered in the training data which has been augmented with named entity information, we add extra copies of that rule to the grammar. We add one copy with all of the named entity information stripped away, and another copy for each other entity type, where the named entity augmentation has been changed to the other entity type.

These additions help, but they are not sufficient. Most entities correspond to noun phrases, so we took all rules which had an NP as a child, and made copies of that rule where the NP was augmented with each possible entity type. These grammar additions sufficed to improve overall performance.

2.3.2 Augmenting the Lexicon

The lexicon is augmented in a similar manner to the rules. For every part of speech tag seen with a named entity annotation, we also add that tag with no named entity information, and a version which has been augmented with each type of named entity.

It would be computationally infeasible to allow any word to have any part of speech tag. We therefore limit the allowed part of speech tags for common words based on the tags they have been observed with in the training data. We also augment each word with a distributional similarity tag, which we discuss in greater depth in Section 3, and allow tags seen with other words which belong to the same distributional similarity cluster. When deciding what tags are allowed for each word, we initially ignore named entity information. Once we determine what base tags are allowed for a word, we also allow that tag, augmented with any type of named entity, if the augmented tag is present in the lexicon.

3 Features

We defined features over both the parse rules and the named entities. Most of our features are over one or the other aspects of the structure, but not both.

Both the named entity and parsing features utilize the words of the sentence, as well as orthographic and distributional similarity information. For each word we computed a *word shape* which encoded

[2] Note that features can include information about other *words*, because the entire sentence is observed. The features *cannot* include information about the labels of those words.

information about capitalization, length, and inclusion of numbers and other non-alphabetic characters. For the distributional similarity information, we had to first train a distributional similarity model. We trained the model described in (Clark, 2000), with code downloaded from his website, on several hundred million words from the British national corpus, and the English Gigaword corpus. The model we trained had 200 clusters, and we used it to assign each word in the training and test data to one of the clusters.

For the named entity features, we used a fairly standard feature set, similar to those described in (Finkel et al., 2005). For parse features, we used the exact same features as described in (Finkel and Manning, 2008). When computing those features, we removed all of the named entity information from the rules, so that these features were just over the parse information and not at all over the named entity information.

Lastly, we have the joint features. We included as features each augmented rule and each augmented label. This allowed the model to learn that certain types of phrasal nodes, such as *NP*s are more likely to be named entities, and that certain entities were more likely to occur in certain contexts and have particular types of internal structure.

4 Data

For our experiments we used the LDC2008T04 OntoNotes Release 2.0 corpus (Hovy et al., 2006). The OntoNotes project leaders describe it as "a large, multilingual richly-annotated corpus constructed at 90% internanotator agreement." The corpus has been annotated with multiple levels of annotation, including constituency trees, predicate structure, word senses, coreference, and named entities. For this work, we focus on the parse trees and named entities. The corpus has English and Chinese portions, and we used only the English portion, which itself has been split into seven sections: ABC, CNN, MNB, NBC, PRI, VOA, and WSJ. These sections represent a mix of speech and newswire data.

4.1 Data Inconsistencies

While other work has utilized the OntoNotes corpus (Pradhan et al., 2007; Yu et al., 2008), this is the first work to our knowledge to simultaneously model the multiple levels of annotation available. Because this is a new corpus, still under development, it is not surprising that we found places where the data was inconsistently annotated, namely with crossing brackets between named entity and tree annotations.

In the places where we found inconsistent annotation it was rarely the case that the different levels of annotation were inherently inconsistent, but rather inconsistency results from somewhat arbitrary choices made by the annotators. For example, when the last word in a sentence ends with a period, such as *Corp.*, one period functions both to mark the abbreviation and the end of the sentence. The convention of the Penn Treebank is to separate the final period and treat it as the end of sentence marker, but when the final word is also part of an entity, that final period was frequently included in the named entity annotation, resulting in the sentence terminating period being part of the entity, and the entity not corresponding to a single phrase. See Figure 2 for an illustration from the data. In this case, we removed the terminating period from the entity, to produce a consistent annotation.

Overall, we found that 656 entities, out of 55,665 total, could not be aligned to a phrase, or multiple contiguous children of a node. We identified and corrected the following sources of inconsistencies:

Periods and abbreviations. This is the problem described above with the *Corp.* example. We corrected it by removing the sentence terminating final period from the entity annotation.

Determiners and PPs. Noun phrases composed of a nested noun phrase and a prepositional phrase were problematic when they also consisted of a determiner followed by an entity. We dealt with this by flattening the nested NP, as illustrated in Figure 3. As we discussed in Section 2.1, this tree will then be augmented with an additional node for the entity (see Figure 1).

Adjectives and PPs. This problem is similar to the previous problem, with the difference being that there are also adjectives preceding the entity. The solution is also similar to the solution to the previous problem. We moved the adjectives from the nested NP into the main NP.

These three modifications to the data solved most, but not all, of the inconsistencies. Another source of problems was conjunctions, such as *North and South Korea*, where *North and South* are a phrase,

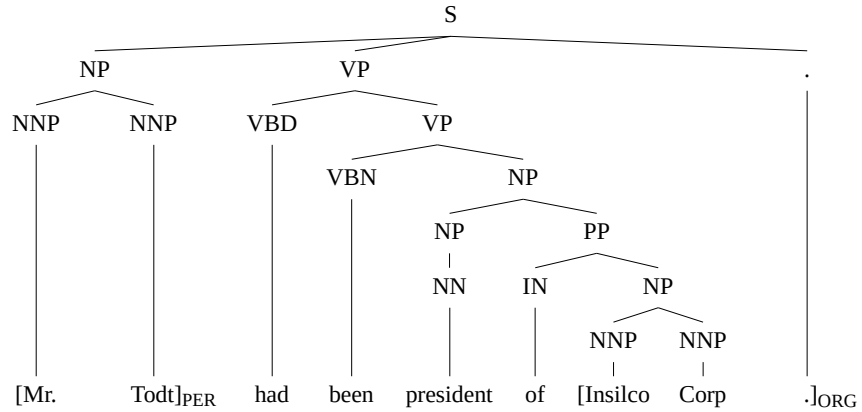


Figure 2: An example from the data of inconsistently labeled named entity and parse structure. The inclusion of the final period in the named entity results in the named entity structure having crossing brackets with the parse structure.

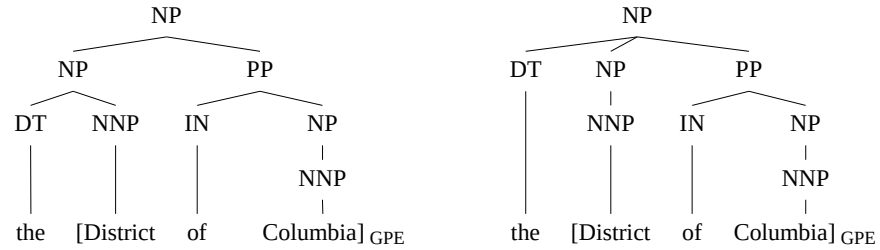


Figure 3: (a) Another example from the data of inconsistently labeled named entity and parse structure. In this instance, we flatten the nested NP, resulting in (b), so that the named entity corresponds to a contiguous set of children of the top-level NP.

but *South Korea* is an entity. The rest of the errors seemed to be due to annotation errors and other random weirdnesses. We ended up unable to make 0.4% of the entities consistent with the parses, so we omitted those entities from the training and test data.

One more change we made to the data was with respect to possessive NPs. When we encountered noun phrases which ended with *(POS 's)* or *(POS ')*, we modified the internal structure of the NP. Originally, these NPs were flat, but we introduced a new nested NP which contained the entire contents of the original NP except for the POS. The original NP label was then changed to PossNP. This change is motivated by the status of *'s* as a phrasal affix or clitic: It is the NP preceding *'s* that is structurally equivalent to other NPs, not the larger unit that includes *'s*. This change has the additional benefit in this context that more named entities will correspond to a single phrase in the parse tree, rather than a contiguous set of phrases.

4.2 Named Entity Types

The data has been annotated with eighteen types of entities. Many of these entity types do not occur very often, and coupled with the relatively small amount of data, make it difficult to learn accurate entity models. Examples are *work of art*, *product*, and *law*. Early experiments showed that it was difficult for even our baseline named entity recognizer, based on a state-of-the-art CRF, to learn these types of entities.[3] As a result, we decided to merge all but the three most dominant entity types into into one general entity type called *misc*. The result was four distinct entity types: *person*, *organization*, *GPE* (geo-political entity, such as a city or a country), and *misc*.

[3]The difficulties were compounded by somewhat inconsistent and occasionally questionable annotations. For example, the word *today* was usually labeled as a *date*, but about 10% of the time it was not labeled as anything. We also found several strange *work of art*s, including *Stanley Cup* and the *U.S.S. Cole*.

	Training		Testing	
	Range	# Sent.	Range	# Sent.
ABC	0–55	1195	56–69	199
CNN	0–375	5092	376–437	1521
MNB	0–17	509	18–25	245
NBC	0–29	552	30–39	149
PRI	0–89	1707	90–112	394
VOA	0–198	1512	199–264	383

Table 1: Training and test set sizes for the six datasets in sentences. The file ranges refer to the numbers within the names of the original OntoNotes files.

5 Experiments

We ran our model on six of the OntoNotes datasets described in Section 4,[4] using sentences of length 40 and under (approximately 200,000 annotated English words, considerably smaller than the Penn Treebank (Marcus et al., 1993)). For each dataset, we aimed for roughly a 75% train / 25% test split. See Table 1 for the the files used to train and test, along with the number of sentences in each.

For comparison, we also trained the parser without the named entity information (and omitted the *NamedEntity* nodes), and a linear chain CRF using just the named entity information. Both the baseline parser and CRF were trained using the exact same features as the joint model, and all were optimized using stochastic gradient descent. The full results can be found in Table 2. Parse trees were scored using *evalB* (the extra *NamedEntity* nodes were ignored when computing evalB for the joint model), and named entities were scored using entity F-measure (as in the CoNLL 2003 *conlleval*).[5]

While the main benefit of our joint model is the ability to get a consistent output over both types of annotations, we also found that modeling the parse and named entities jointly resulted in improved performance on both. When looking at these numbers, it is important to keep in mind that the sizes of the training and test sets are significantly smaller than the Penn Treebank. The largest of the six datasets, CNN, has about one seventh the amount of training data as the Penn Treebank, and the smallest, MNB, has around 500 sentences from which to train. Parse performance was improved by the joint model for five of the six datasets, by up to 1.36%. Looking at the parsing improvements on a per-label basis, the largest gains came from improved identication of NML consituents, from an F-score of 45.9% to 57.0% (on all the data combined, for a total of 420 NML constituents). This label was added in the new treebank annotation conventions, so as to identify internal left-branching structure inside previously flat NPs. To our surprise, performance on NPs only increased by 1%, though over 12,949 constituents, for the largest improvement in absolute terms. The second largest gain was on PPs, where we improved by 1.7% over 3,775 constituents. We tested the significance of our results (on all the data combined) using Dan Bikel's randomized parsing evaluation comparator[6] and found that both the precision and recall gains were significant at $p \leq 0.01$.

Much greater improvements in performance were seen on named entity recognition, where most of the domains saw improvements in the range of 3–4%, with performance on the *VOA* data improving by nearly 9%, which is a 45% reduction in error. There was no clear trend in terms of precision versus recall, or the different entity types. The first place to look for improvements is with the boundaries for named entities. Once again looking at all of the data combined, in the baseline model there were 203 entities where part of the entity was found, but one or both boundaries were incorrectly identified. The joint model corrected 72 of those entities, while incorrectly identifying the boundaries of 37 entities which had previously been correctly identified. In the baseline NER model, there were 243 entities for which the boundaries were correctly identified, but the type of entity was incorrect. The joint model corrected 80 of them, while changing the labels of 39 entities which had previously been correctly identified. Additionally, 190 entities were found which the baseline model had missed entirely, and 68 enti-

[4] These datasets all consistently use the new conventions for treebank annotation, while the seventh WSJ portion is currently still annotated in the original 1990s style, and so we left the WSJ portion aside.

[5] Sometimes the parser would be unable to parse a sentence (less than 2% of sentences), due to restrictions in part of speech tags. Because the underlying grammar (ignoring the additional named entity information) was the same for both the joint and baseline parsers, it is the case that whenever a sentence is unparseable by either the baseline or joint parser it is in fact unparsable by both of them, and would affect the parse scores of both models equally. However, the CRF is able to named entity tag any sentence, so these unparsable sentences had an effect on the named entity score. To combat this, we fell back on the baseline CRF model to get named entity tags for unparsable sentences.

[6] Available at http://www.cis.upenn.edu/ dbikel/software.html

		Parse Labeled Bracketing			Named Entities			Training
		Precision	Recall	F_1	Precision	Recall	F_1	Time
ABC	Just Parse	70.18%	70.12%	**70.15%**		–		25m
	Just NER		–		76.84%	72.32%	74.51%	
	Joint Model	69.76%	70.23%	69.99%	77.70%	72.32%	**74.91%**	45m
CNN	Just Parse	76.92%	77.14%	77.03%		–		16.5h
	Just NER		–		75.56%	76.00%	75.78%	
	Joint Model	77.43%	77.99%	**77.71%**	78.73%	78.67%	**78.70%**	31.7h
MNB	Just Parse	63.97%	67.07%	65.49%		–		12m
	Just NER		–		72.30%	54.59%	62.21%	
	Joint Model	63.82$	67.46%	**65.59%**	71.35%	62.24%	**66.49%**	19m
NBC	Just Parse	59.72%	63.67%	61.63%		–		10m
	Just NER		–		67.53%	60.65%	63.90%	
	Joint Model	60.69%	65.34%	**62.93%**	71.43%	64.81%	**67.96%**	17m
PRI	Just Parse	76.22%	76.49%	76.35%		–		2.4h
	Just NER		–		82.07%	84.86%	83.44%	
	Joint Model	76.88%	77.95%	**77.41%**	86.13%	86.56%	**86.34%**	4.2h
VOA	Just Parse	76.56%	75.74%	76.15%		–		2.3h
	Just NER		–		82.79%	75.96%	79.23%	
	Joint Model	77.58%	77.45%	**77.51%**	88.37%	87.98%	**88.18%**	4.4h

Table 2: Full parse and NER results for the six datasets. Parse trees were evaluated using evalB, and named entities were scored using macro-averaged F-measure (conlleval).

ties were lost. We tested the statistical significance of the gains (of all the data combined) using the same sentence-level, stratified shuffling technique as Bikel's parse comparator and found that both precision and recall gains were significant at $p < 10^{-4}$.

An example from the data where the joint model helped improve both parse structure and named entity recognition is shown in Figure 4. The output from the individual models is shown in part (a), with the output from the named entity recognizer shown in brackets on the words at leaves of the parse. The output from the joint model is shown in part (b), with the named entity information encoded within the parse. In this example, the named entity *Egyptian Islamic Jihad* helped the parser to get its surrounding context correct, because it is improbable to attach a PP headed by *with* to an *organization*. At the same time, the surrounding context helped the joint model correctly identify *Egyptian Islamic Jihad* as an *organization* and not a *person*. The baseline parser also incorrectly added an extra level of structure to the person name *Osama Bin Laden*, while the joint model found the correct structure.

6 Related Work

A pioneering antecedent for our work is (Miller et al., 2000), who trained a Collins-style generative parser (Collins, 1997) over a syntactic structure augmented with the *template entity* and *template relations* annotations for the MUC-7 shared task. Their sentence augmentations were similar to ours, but they did not make use of features due to the generative nature of their model. This approach was not followed up on in other work, presumably because around this time nearly all the activity in named entity and relation extraction moved to the use of discriminative sequence models, which allowed the flexible specification of feature templates that are very useful for these tasks. The present model is able to bring together both these lines of work, by integrating the strengths of both approaches.

There have been other attempts in NLP to jointly model multiple levels of structure, with varying degrees of success. Most work on joint parsing and semantic role labeling (SRL) has been disappointing, despite obvious connections between the two tasks. Sutton and McCallum (2005) attempted to jointly model PCFG parsing and SRL for the CoNLL 2005 shared task, but were unable to improve performance on either task. The CoNLL 2008 shared task (Surdeanu et al., 2008) was joint dependency parsing and SRL, but the top performing systems decoupled the tasks, rather than building joint models. Zhang and Clark (2008) successfully built a joint

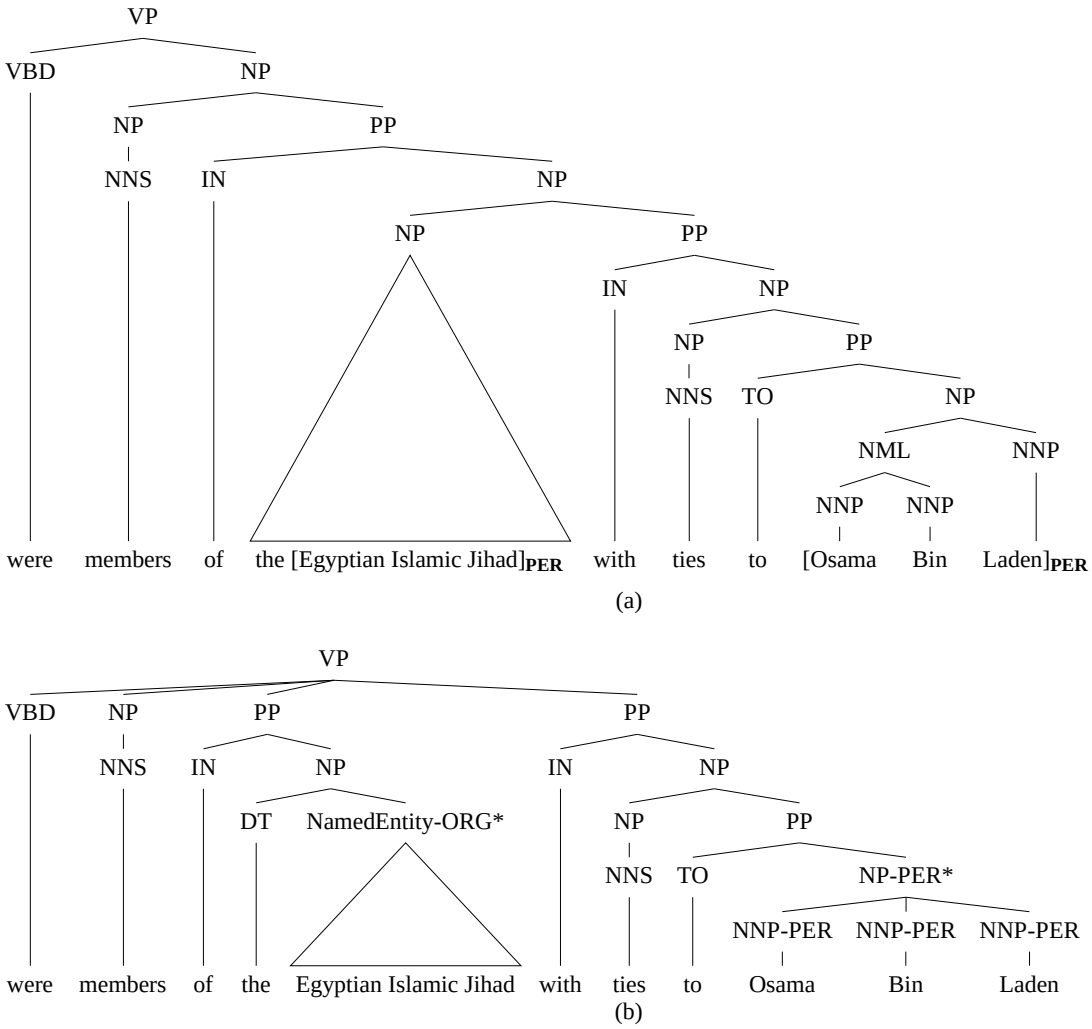


Figure 4: An example for which the joint model helped with both parse structure and named entity recognition. The individual models (a) incorrectly attach the PP, label *Egyptian Islamic Jihad* as a *person*, and incorrectly add extra internal structure to *Osama Bin Laden*. The joint model (b) gets both the structure and the named entity correct.

model of Chinese word segmentation and parts of speech using a single perceptron.

An alternative approach to joint modeling is to take a pipelined approach. Previous work on linguistic annotation pipelines (Finkel et al., 2006; Hollingshead and Roark, 2007) has enforced consistency from one stage to the next. However, these models are only used at test time; training of the components is still independent. These models also have the potential to suffer from search errors and are not guaranteed to find the optimal output.

7 Conclusion

We presented a discriminatively trained joint model of parsing and named entity recognition, which improved performance on both tasks. Our model is based on a discriminative constituency parser, with the data, grammar, and features carefully constructed for the joint task. In the future, we would like to add other levels of annotation available in the OntoNotes corpus to our model, including word sense disambiguation and semantic role labeling.

Acknowledgements

The first author is supported by a Stanford Graduate Fellowship. This paper is based on work funded in part by the Defense Advanced Research Projects Agency through IBM. The content does not necessarily reflect the views of the U.S. Government, and no official endorsement should be inferred. We also wish to thank the creators of OntoNotes, without which this project would not have been possible.

References

Alexander Clark. 2000. Inducing syntactic categories by context distribution clustering. In *Proc. of Conference on Computational Natural Language Learning*, pages 91–94, Lisbon, Portugal.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *ACL 1997*.

Jenny Rose Finkel and Christopher D. Manning. 2008. Efficient, feature-based conditional random field parsing. In *ACL/HLT-2008*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005*.

Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *EMNLP 2006*.

Kristy Hollingshead and Brian Roark. 2007. Pipeline iteration. In *ACL 2007*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *HLT-NAACL 2006*.

Jin-Dong Kim, Tomoko Ohta, Yuka Teteisi, and Jun'ichi Tsujii. 2003. Genia corpus – a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl. 1):i180–i182.

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

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *In 6th Applied Natural Language Processing Conference*, pages 226–233.

Andrew Ng and Michael Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems (NIPS)*.

Sameer S. Pradhan, Lance Ramshaw, Ralph Weischedel, Jessica MacBride, and Linnea Micciulla. 2007. Unrestricted coreference: Identifying entities and events in ontonotes. *International Conference on Semantic Computing*, 0:446–453.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL)*, Manchester, UK.

Charles Sutton and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *Conference on Natural Language Learning (CoNLL)*.

V. N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley & Sons.

Liang-Chih Yu, Chung-Hsien Wu, and Eduard Hovy. 2008. OntoNotes: Corpus cleanup of mistaken agreement using word sense disambiguation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1057–1064.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *ACL 2008*.

Minimal-length linearizations for mildly context-sensitive dependency trees

Y. Albert Park
Department of Computer Science and Engineering
9500 Gilman Drive
La Jolla, CA 92037-404, USA
yapark@ucsd.edu

Roger Levy
Department of Linguistics
9500 Gilman Drive
La Jolla, CA 92037-108, USA
rlevy@ling.ucsd.edu

Abstract

The extent to which the organization of natural language grammars reflects a drive to minimize dependency length remains little explored. We present the first algorithm polynomial-time in sentence length for obtaining the minimal-length linearization of a dependency tree subject to constraints of mild context sensitivity. For the minimally context-sensitive case of gap-degree 1 dependency trees, we prove several properties of minimal-length linearizations which allow us to improve the efficiency of our algorithm to the point that it can be used on most naturally-occurring sentences. We use the algorithm to compare optimal, observed, and random sentence dependency length for both surface and deep dependencies in English and German. We find in both languages that analyses of surface and deep dependencies yield highly similar results, and that mild context-sensitivity affords very little reduction in minimal dependency length over fully projective linearizations; but that observed linearizations in German are much closer to random and farther from minimal-length linearizations than in English.

1 Introduction

This paper takes up the relationship between two hallmarks of natural language dependency structure. First, there seem to be qualitative constraints on the relationship between the dependency structure of the words in a sentence and their linear ordering. In particular, this relationship seems to be such that any natural language sentence, together with its dependency structure, should be generable by a *mildly context-sensitivity* formalism (Joshi, 1985), in particular a linear context-free rewrite system in which the right-hand side of each rule has a distinguished head (Pollard, 1984; Vijay-Shanker et al., 1987; Kuhlmann, 2007). This condition places strong constraints on the linear contiguity of word-word dependency relations, such that only limited classes of crossing context-free dependency structures may be admitted.

The second constraint is a softer preference for words in a dependency relation to occur in close proximity to one another. This constraint is perhaps best documented in psycholinguistic work suggesting that large distances between governors and dependents induce processing difficulty in both comprehension and production (Hawkins, 1994, 2004; Gibson, 1998; Jaeger, 2006). Intuitively there is a relationship between these two constraints: consistently large dependency distances in a sentence would require many crossing dependencies. However, it is not the case that crossing dependencies always mean longer dependency distances. For example, (1) below has no crossing dependencies, but the distance between *arrived* and its dependent *Yesterday* is large. The overall dependency length of the sentence can be reduced by extraposing the relative clause *who was wearing a hat*, resulting in (2), in which the dependency *Yesterday*→*arrived* crosses the dependency *woman*←*who*.

(1) Yesterday *a woman who was wearing a hat* arrived.

(2) Yesterday *a woman* arrived *who was wearing a hat*.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 335–343,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

There has been some recent work on dependency length minimization in natural language sentences (Gildea and Temperley, 2007), but the relationship between the precise constraints on available linearizations and dependency length minimization remains little explored. In this paper, we introduce the first efficient algorithm for obtaining linearizations of dependency trees that minimize overall dependency lengths subject to the constraint of mild context-sensitivity, and use it to investigate the relationship between this constraint and the distribution of dependency length actually observed in natural languages.

2 Projective and mildly non-projective dependency-tree linearizations

In the last few years there has been a resurgence of interest in computation on dependency-tree structures for natural language sentences, spurred by work such as McDonald et al. (2005a,b) showing that working with dependency-tree syntactic representations in which each word in the sentence corresponds to a node in the dependency tree (and vice versa) can lead to algorithmic benefits over constituency-structure representations. The *linearization* of a dependency tree is simply the linear order in which the nodes of the tree occur in a surface string. There is a broad division between two classes of linearizations: *projective* linearizations that do not lead to any crossing dependencies in the tree, and *non-projective* linearizations that involve at least one crossing dependency pair. Example (1), for example, is projective, whereas Example (2) is non-projective due to the crossing between the *Yesterday→arrived* and *woman←who* dependencies.

Beyond this dichotomy, however, the homomorphism from headed tree structures to dependency structures (Miller, 2000) can be used together with work on the mildly context-sensitive formalism linear context-free rewrite systems (LCFRSs) (Vijay-Shanker et al., 1987) to characterize various classes of *mildly non-projective* dependency-tree linearizations (Kuhlmann and Nivre, 2006). The LCFRSs are an infinite sequence of classes of formalism for generating surface strings through derivation trees in a rule-based context-free rewriting system. The i-th LCFRS class (for $i = 0, 1, 2, \dots$) imposes the constraint that every node in the derivation tree maps to to a collection of at most $i+1$ contiguous substrings. The 0-th class of LCFRS, for example, corresponds to the context-free grammars, since each node in the derivation tree must map to a single contiguous substring; the 1st class of LCFRS corresponds to Tree-Adjoining Grammars (Joshi et al., 1975), in which each node in the derivation tree must map to at most a pair of contiguous substrings; and so forth. The dependency trees induced when each rewrite rule in an i-th order LCFRS distinguish a unique *head* can similarly be characterized by being of *gap-degree* i, so that i is the maximum number of gaps that may appear between contiguous substrings of any subtree in the dependency tree (Kuhlmann and Möhl, 2007). The dependency tree for Example (2), for example, is of gap-degree 1. Although there are numerous documented cases in which projectivity is violated in natural language, there are exceedingly few documented cases in which the documented gap degree exceeds 1 (though see, for example, Kobele, 2006).

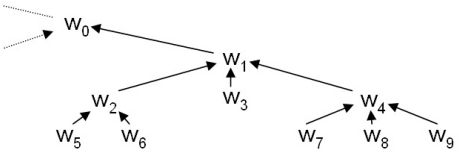


Figure 1: Sample dependency subtree for Figure 2

3 Finding minimal dependency-length linearizations

Even under the strongest constraint of projectivity, the number of possible linearizations of a dependency tree is exponential in both sentence length and arity (the maximum number of dependencies for any word). As pointed out by Gildea and Temperley (2007), however, finding the unconstrained minimal-length linearization is a well-studied problem with an $O(n^{1.6})$ solution (Chung, 1984). However, this approach does not take into account constraints of projectivity or mild context-sensitivity.

Gildea and Temperley themselves introduced a novel efficient algorithm for finding the minimized dependency length of a sentence subject to the constraint that the linearization is projective. Their algorithm can perhaps be most simply understood by making three observations. First, the total depen-

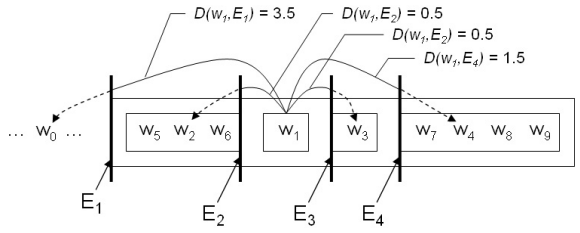


Figure 2: Dependency length factorization for efficient projective linearization, using the dependency subtree of Figure 1

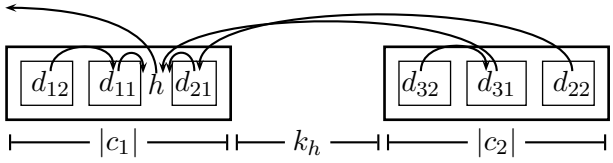


Figure 3: Factorizing dependency length at node w_i of a mildly context-sensitive dependency tree. This partial linearization of head with dependent components makes c_1 the head component and leads to $l = 2$ links crossing between c_1 and c_2.

dency length of a projective linearization can be written as

$$\sum_{w_i} \left[D(w_i, E_i) + \sum_{w_j \overset{dep}{\rightarrow} w_i} D(w_i, E_j) \right] \quad (1)$$

where E_i is the boundary of the contiguous substring corresponding to the dependency subtree rooted at w_i which stands between w_i and its governor, and $D(w_i, E_j)$ is the distance from w_i to E_j, with the special case of $D(w_{root}, E_{root}) = 0$ (Figures 1 and 2). Writing the total dependency length this way makes it clear that each term in the outer sum can be optimized independently, and thus one can use dynamic programming to recursively find optimal subtree orderings from the bottom up. Second, for each subtree, the optimal ordering can be obtained by placing dependent subtrees on alternating sides of *w* from inside out in order of increasing length. Third, the total dependency lengths between any words withing an ordering stays the same when the ordering is reversed, letting us assume that $D(w_i, E_i)$ will be the length to the closest edge. These three observations lead to an algorithm with worst-case complexity of $O(n \log m)$ time, where n is sentence length and m is sentence arity. (The $\log m$ term arises from the need to sort the daughters of each node into descending order of length.)

When limited subclasses of nonprojectivity are admitted, however, the problem becomes more difficult because total dependency length can no longer be written in such a simple form as in Equation (1). Intuitively, the size of the effect on dependency length of a decision to order a given subtree discontiguously, as in *a woman… who was wearing a hat* in Example (2), cannot be calculated without consulting the length of the string that the discontiguous subtree would be wrapped around. Nevertheless, for any limited gap degree, it is possible to use a different factorization of dependency length that keeps computation polynomial in sentence length. We introduce this factorization in the next section.

4 Minimization with limited gap degree

We begin by defining some terms. We use the word *component* to refer to a full linearization of a subtree in the case where it is realized as a single contiguous string, or to refer to any of of the contiguous substrings produced when a subtree is realized discontiguously. We illustrate the factorization for gap-degree 1, so that any subtree has at most two components. We refer to the component containing the head of the subtree as the *head component*, the remaining component as the *dependent component*, and for any given (head component, dependent component) pair, we use *pair component* to refer to the other component in the pair. We refer to the two components of dependent d_j as d_{j1} and d_{j2} respectively, and assume that d_{j1} is the head component. When dependencies can cross, total dependency length cannot be factorized as simply as in Equation (1) for the projective case. However, we can still make use of a more complex factorization of the total dependency length as follows:

$$\sum_{w_i} \left[D(w_i, E_i) + \sum_{w_j \overset{\text{dep}}{\rightarrow} w_i} \left[D(w_i, E_j) + l_j k_j \right] \right] \quad (2)$$

where l_j is the number of links crossing between the two components of d_j, and k_j is the distance added between these two components by the partial linearization at w_i. Figure 3 illustrates an example of

such a partial linearization, where k_2 is $|d_{31}| + |d_{32}|$ due to the fact that the links between d_{21} and d_{22} have to cross both components of d_3. The factorization in Equation (2) allows us to use dynamic programming to find minimal-length linearizations, so that worst-case complexity is polynomial rather than exponential in sentence length. However, the additional term in the factorization means that we need to track the number of links l crossing between the two components of the subtree S_i headed by w_i and the component lengths $|c_1|$ and $|c_2|$. Additionally, the presence of crossing dependencies means that Gildea and Temperley's proof that ordering dependent components from the inside out in order of increasing length no longer goes through. This means that at each node w_i we need to hold on to the minimal-length partial linearization for each combination of the following quantities:

- $|c_2|$ (which also determines $|c_1|$);
- the number of links l between c_1 and c_2;
- and the direction of the link between w_i and its governor.

We shall refer to a combination of these factors as a *status set*. The remainder of this section describes a dynamic-programming algorithm for finding optimal linearizations based on the factorization in Equation (2), and continues with several further findings leading to optimizations that make the algorithm tractable for naturally occurring sentences.

4.1 Algorithm 1

Our first algorithm takes a tree and recursively finds the optimal orderings for each possible status set of each of its child subtrees, which it then uses to calculate the optimal ordering of the tree. To calculate the optimal orderings for each possible status set of a subtree S, we use the brute-force method of choosing all combinations of one status set from each child subtree, and for each combination, we try all possible orderings of the components of the child subtrees, calculate all possible status sets for S, and store the minimal dependency value for each appearing status set of S. The number of possible length pairings $|c_1|, |c_2|$ and number of crossing links l are each bounded above by the sentence length n, so that the maximum number of status sets at each node is bounded above by n^2. Since the sum of the status sets of all child subtrees is also bounded by n^2, the maximum number of status set combinations is bounded by $(\frac{n^2}{m})^m$ (obtainable from the inequality of arithmetic and geometric means). There are $(2m+1)!m$ possible arrangements of head word and dependent components into two components. Since there are n nodes in the tree and each possible combination of status sets from each dependent sub tree must be tried, this algorithm has worst-case complexity of $O((2m+1)!mn(\frac{n^2}{m})^m)$. This algorithm could be generalized for mildly context-sensitive linearizations polynomial in sentence length for any gap degree desired, by introducing additional l terms denoting the number of links between pairs of components. However, even for gap degree 1 this bound is incredibly large, and as we show in Figure 7, algorithm 1 is not computationally feasible for batch processing sentences of arity greater than 5.

4.2 Algorithm 2

We now show how to speed up our algorithm by proving by contradiction that for any optimal ordering which minimizes the total dependency length with the two-cluster constraint, for any given subtree S and its child subtree C, the pair components c_1 and c_2 of a child subtree C must be placed on opposite sides of the head h of subtree S.

Let us assume that for some dependency tree structure, there exists an optimal ordering where c_1 and c_2 are on the same side of h. Let us refer to the ordered set of words between c_1 and c_2 as v. None of the words in v will have dependency links to any of the words in c_1 and c_2, since the dependencies of the words in c_1 and c_2 are either between themselves or the one link to h, which is not between the two components by our assumption. There will be $j_1 \geq 0$ links from v going over c_1, $j_2 \geq 0$ dependency links from v going over c_2, and $l \geq 1$ links between c_1 and c_2. Without loss of generality, let us assume that h is on the right side of c_2. Let us consider the effect on total dependency length of swapping c_1 with v, so that the linear ordering is $v\ c_1\ c_2 \prec h$. The total dependency length of the new word ordering changes by $-j_1|c_1| - l|v| + j_2|c_1|$ if c_2 is the head component, and decreases by another $|v|$ if c_1 is the head component. Thus the total change in dependency length

is less than or equal to

$$(j_2 - j_1)|c_1| - l \times |v| < (j_2 - j_1)|c_1| \quad (3)$$

If instead we swap places of v with c_2 instead of c_1 so that we have $c_1\ c_2\ v \prec h$, we find that the total change in dependency length is less than or equal to

$$(j_1 - j_2)|c_2| - (l-1)|v| \leq (j_1 - j_2)|c_2| \quad (4)$$

It is impossible for the right-hand sides of (3) and (4) to be positive at the same time, so swapping v with either c_1 or c_2 must lead to a linearization with lower overall dependency length. But this is a contradiction to our original assumption, so we see that for any optimal ordering, all split child subtree components c_1 and c_2 of the child subtree of S must be placed on opposite sides of the head h.

This constraint allows us to simplify our algorithm for finding the minimal-length linearization. Instead of going through all logically possible orderings of components of the child subtrees, we can now decide on which side the head component will be on, and go through all possible orderings for each side. This changes the factorial part of our algorithm run time from $(2m+1)!m$ to $2^m(m!)^2m$, giving us $O(2^m(m!)^2mn(\frac{n^2}{m})^m)$, greatly reducing actual processing time.

4.3 Algorithm 3

We now present two more findings for further increasing the efficiency of the algorithm. First, we look at the status sets which need to be stored for the dynamic programming algorithm. In the straightforward approach we first presented, we stored the optimal dependency lengths for all cases of possible status sets. We now know that we only need to consider cases where the pair components are on opposite sides. This means the direction of the link from the head to the parent will always be toward the inside direction of the pair components, so we can re-define the status set as (p, l) where p is again the length of the dependent component, and l is the number of links between the two pair components. If the p values for sets s_1 and s_2 are equal, s_1 has a smaller number of links than s_2 ($l_{s_1} \leq l_{s_2}$) and s_1 has a smaller or equal total dependency length to s_2, then replacing the components of s_2 with s_1 will always give us the same or more optimal total dependency length. Thus, we do not have to store instances of these cases for our algorithm.

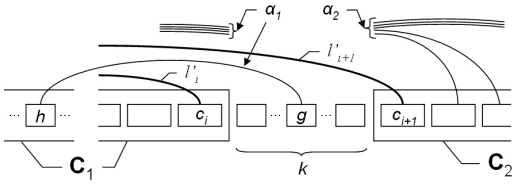


Figure 4: Initial setup for latter part of optimization proof in section 4.4. To the far left is the head h of subtree S. The component pair C_1 and C_2 makes up S, and g is the governor of h. The length of the substring v between C_1 and C_2 is k. c_i and c_{i+1} are child subtree components.

Next, we prove by contradiction that for any two status sets s_1 and s_2, if $p_{s_1} > p_{s_2} > 0$, $l_{s_1} = l_{s_2}$, and the TOTAL INTERNAL DEPENDENCY LENGTH t_1 of s_1—defined as the sum in Equation (2) over only those words inside the subtree headed by h—is less than or equal to t_2 of s_2, then using s_1 will be at least as good as s_2, so we can ignore s_2. Let us suppose that the optimal linearization can use s_2 but not s_1. Then in the optimal linearization, the two pair components $c_{s_2,1}$ and $c_{s_2,2}$ of s_2 are on opposite sides of the parent head h. WLOG, let us assume that components $c_{s_1,1}$ and $c_{s_2,1}$ are the dependent components. Let us denote the total number of links going over $c_{s_2,1}$ as j_1 and the words between $c_{s_2,1}$ and $c_{s_2,2}$ as v (note that v must contain h). If we swap $c_{s_2,1}$ with v, so that $c_{s_2,1}$ lies adjacent to $c_{s_2,2}$, then there would be j_2+1 links going over $c_{s_2,1}$. By moving $c_{s_2,1}$ from opposite sides of the head to be right next to $c_{s_2,2}$, the total dependency length of the sentence changes by $-j_1|c_{s_2,1}| - l_{s_2}|v| + (j_2+1)|c_{s_2,1}|$. Since the ordering was optimal, we know that

$$(j_2 - j_1 + 1)|c_{s_2,1}| - l_{s_2}|v| \geq 0$$

Since $l > 0$, we can see that $j_1 - j_2 \leq 0$. Now, instead of swapping v with $c_{s_2,1}$, let us try substituting the components from s_1 instead of s_2. The change of the total dependency length of the sentence will be:

$$\begin{aligned} & j_1 \times (|c_{s_1,1}| - |c_{s_2,1}|) + j_2 \times (|c_{s_1,2}| \\ & \quad -|c_{s_2,2}|) + t_1 - t_2 \\ = \ & (j_1 - j_2) \times (p_{s_1} - p_{s_2}) + (t_1 - t_2) \end{aligned}$$

Since $j_1 - j_2 \leq 0$ and $p_{s_1} > p_{s_2}$, the first term is less than or equal to 0 and since $t_1 - t_2 \leq 0$, the total dependency length will have been be equal or

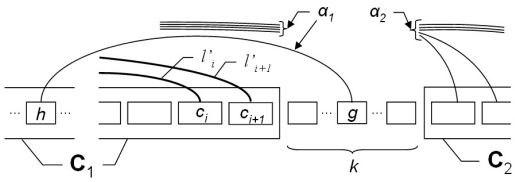


Figure 5: Moving c_{i+1} to C_1

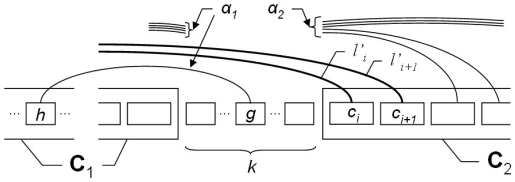


Figure 6: Moving c_i to C_2

have decreased. But this contradicts our assumption that only s_2 can be part of an optimal ordering.

This finding greatly reduces the number of status sets we need to store and check higher up in the algorithm. The worst-case complexity remains $O(2^m m!^2 mn(\frac{n^2}{m})^m)$, but the actual runtime is reduced by several orders of magnitude.

4.4 Algorithm 4

Our last optimization is on the ordering among the child subtree components on each side of the subtree head h. The initially proposed algorithm went through all combinations of possible orderings to find the optimal dependency length for each status set. By the first optimization in section 4.2 we have shown that we only need to consider the orderings in which the components are on opposite sides of the head. We now look into the ordering of the components on each side of the head. We first define the *rank value* r for each component c as follows:

$$\frac{|c|}{\text{\# links between } c \text{ and its pair component} + I(c)}$$

where $I(c)$ is the indicator function having value 1 if c is a head component and 0 otherwise . Using this definition, we prove by contradiction that the ordering of the components from the head outward must be in order of increasing rank value.

Let us suppose that at some subtree S headed by h and with head component C_1 and dependent component C_2, there is an optimal linearization in which there exist two components c_i and c_{i+1} of immediate subtrees of S such that c_i is closer to h, the components have rank values r_i and r_{i+1} respectively, $r_i > r_{i+1}$, and no other component of the immediate subtrees of S intervenes between c_i and c_{i+1}. We shall denote the number of links between each component and its pair component as l_i, l_{i+1}. Let $l'_i = l_i + I(c_i)$ and $l'_{i+1} = l_{i+1} + I(c_{i+1})$. There are two cases to consider: either (1) c_i and c_{i+1} are within the same component of S, or (2) c_i is at the edge of C_1 nearest C_2 and c_{i+1} is at the edge of C_2 neareast C_1.

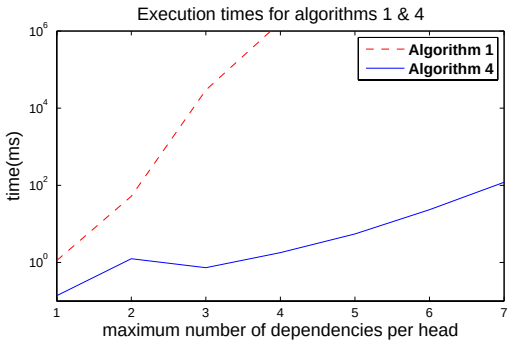


Figure 7: Timing comparison of first and fully optimized algorithms

Consider case 1, and let us swap c_i with c_{i+1}; this affects only the lengths of links involving connections to c_i or c_{i+1}. The total dependency length of the new linearization will change by

$$-l'_{i+1}|c_i| + l'_i|c_{i+1}| = -l'_i l'_{i+1}(r_i - r_{i+1}) < 0$$

This is a contradiction to the assumption that we had an optimal ordering.

Now consider case 2, which is illustrated in Figure 4. We denote the number of links going over c_i and c_{i+1}, excluding links to c_i, c_{i+1} as α_1 and α_2 respectively, and the length of words between the edges of C_1 and C_2 as k. Let us move c_{i+1} to the outermost position of C_1, as shown in Figure 5. Since the original linearization was optimal, we have:

$$\begin{aligned} -\alpha_2|c_{i+1}| + \alpha_1|c_{i+1}| - l'_{i+1}k &\geq 0 \\ (\alpha_1 - \alpha_2)|c_{i+1}| &\geq l'_{i+1}k \\ (\alpha_1 - \alpha_2)r_{i+1} &\geq k \end{aligned}$$

Let us also consider the opposite case of moving c_i to the inner edge of C_2, as shown in Figure 6. Once again due to optimality of the original linearization, we have

DLA	English		German	
	Surface	Deep	Surface	Deep
Optimal with one crossing dependency	32.7	33.0	24.5	23.3
Optimal with projectivity constraint	34.1	34.4	25.5	24.2
Observed	46.6	48.0	43.6	43.1
Random with projectivity constraint	82.4	82.8	50.6	49.2
Random with two-cluster constraint	84.0	84.3	50.7	49.5
Random ordering with no constraint	183.2	184.2	106.9	101.1

Table 1: Average sentence dependency lengths(with max arity of 10)

$$-\alpha_1|c_i| + \alpha_2|c_i| + l'_i k \geq 0$$
$$(\alpha_2 - \alpha_1)|c_i| \geq -l'_i k$$
$$(\alpha_1 - \alpha_2)r_i \leq k$$

But this is a contradiction, since $r_i > r_{i+1}$. Combining the two cases, we can see that regardless of where the components may be split, in an optimal ordering the components going outwards from the head must have an increasing rank value.

This result allows us to simplify our algorithm greatly, because we no longer need to go through all combinations of orderings. Once it has been decided which components will come on each side of the head, we can sort the components by rank value and place them from the head out. This reduces the factorial component of the algorithm's complexity to $m \log m$, and the overall worst-case complexity to $O(nm^2 \log m(\frac{2n^2}{m})^m)$. Although this is still exponential in the arity of the tree, nearly all sentences encountered in treebanks have an arity low enough to make the algorithm tractable and even very efficient, as we show in the following section.

5 Empirical results

Using the above algorithm, we calculated minimal dependency lengths for English sentences from the WSJ portion of the Penn Treebank, and for German sentences from the NEGRA corpus. The English-German comparison is of interest because word order is freer, and crossing dependencies more common, in German than in English (Kruijff and Vasishth, 2003). We extracted dependency trees from these corpora using the head rules of Collins (1999) for English, and the head rules of Levy and Manning (2004) for German. Two dependency trees were extracted from each sentence, the *surface tree* extracted by using the head rules on the context-free tree representation (i.e. no crossing dependencies), and the *deep tree* extracted by first returning discontinuous dependents (marked by *T* and *ICH* in WSJ, and by *T* in the Penn-format version of NEGRA) before applying head rules. Figure 7 shows the average time it takes to calculate the minimal dependency length with crossing dependencies for WSJ sentences using the unoptimized algorithm of Section 4.1 and the fully optimized algorithm of Section 4.4. Timing tests were implemented and performed using Java 1.6.0_10 on a system running Linux 2.6.18-6-amd64 with a 2.0 GHz Intel Xeon processor and 16 gigs of memory, run on a single core. We can see from Figure 7 that the straight-forward dynamic programming algorithm takes many more magnitudes of time than our optimized algorithm, making it infeasible to calculate the minimal dependency length for larger sentences. The results we present below were obtained with the fully optimized algorithm from the sentences with a maximum arity of 10, using 49,176 of the 49,208 WSJ sentences and 20,563 of the 20,602 NEGRA sentences.

Summary results over all sentences from each corpus are shown in Table 1. We can see that for both corpora, the oberved dependency length is smaller than the dependency length of random orderings, even when the random ordering is subject to the projectivity constraint. Relaxing the projectivity constraint by allowing crossing dependencies introduces a slightly lower optimal dependency length. The average sentence dependency lengths for the three random orderings are significantly higher than the observed values. It is interesting to note that the random orderings given the projectivity constraint and the two-cluster constraint have very similar dependency lengths, where as a total random ordering

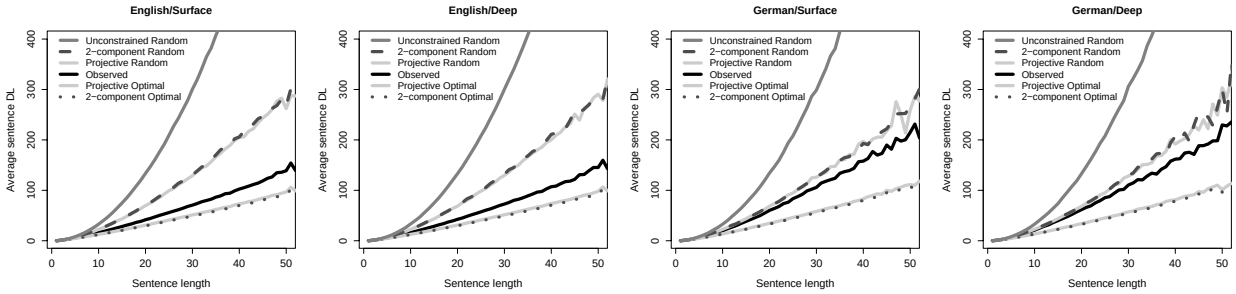


Figure 8: Average sentence DL as a function of sentence length. Legend is ordered top curve to bottom curve.

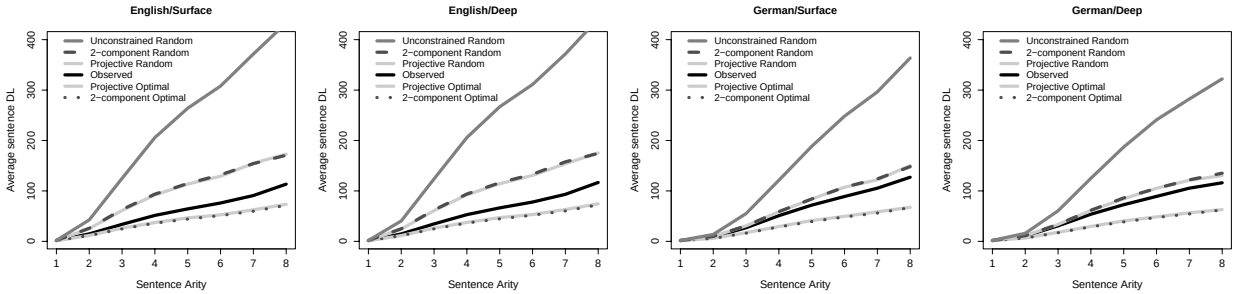


Figure 9: Average sentence DL as a function of sentence arity. Legend is ordered top curve to bottom curve.

increases the dependency length significantly.

NEGRA generally has shorter sentences than WSJ, so we need a more detailed picture of dependency length as a function of sentence length; this is shown in Figure 8. As in Table 1, we see that English, which has less crossing dependency structures than German, has observed DL closer to optimal DL and farther from random DL. We also see that the random and observed DLs behave very similarly across different sentence lengths in English and German, but observed DL grows faster in German. Perhaps surprisingly, optimal projective DL and gap-degree 1 DL tend to be very similar even for longer sentences. The picture as a function of sentence arity is largely the same (Figure 9).

6 Conclusion

In this paper, we have presented an efficient dynamic programming algorithm which finds minimum-length dependency-tree linearizations subject to constraints of mild context-sensitivity. For the gap-degree 1 case, we have proven several properties of these linearizations, and have used these properties to optimize our algorithm. This made it possible to find minimal dependency lengths for sentences from the English Penn Treebank WSJ and German NEGRA corpora. The results show that for both languages, using surface dependencies and deep dependencies lead to generally similar conclusions, but that minimal lengths for deep dependencies are consistently slightly higher for English and slightly lower for German. This may be because German has many more crossing dependencies than English. Another finding is that the difference between average sentence DL does not change much between optimizing for the projectivity constraint and the two-cluster constraint: projectivity seems to give natural language almost all the flexibility it needs to minimize DL. For both languages, the observed linearization is much closer in DL to optimal linearizations than to random linearizations; but crucially, we see that English is closer to the optimal linearization and farther from random linearization than German. This finding is resonant with the fact that German has richer morphology and overall greater variability in observed word order, and with psycholinguistic results suggesting that dependencies of greater linear distance do not always pose the same increased processing load in German sentence comprehension as they do in English (Konieczny, 2000).

References

Chung, F. R. K. (1984). On optimal linear arrangements of trees. *Computers and Mathematics with Applications*, 10:43–60.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

Gibson, E. (1998). Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68:1–76.

Gildea, D. and Temperley, D. (2007). Optimizing grammars for minimum dependency length. In *Proceedings of ACL*.

Hawkins, J. A. (1994). *A Performance Theory of Order and Constituency*. Cambridge.

Hawkins, J. A. (2004). *Efficiency and Complexity in Grammars*. Oxford University Press.

Jaeger, T. F. (2006). *Redundancy and Syntactic Reduction in Spontaneous Speech*. PhD thesis, Stanford University, Stanford, CA.

Joshi, A. K. (1985). How much context-sensitivity is necessary for characterizing structural descriptions – Tree Adjoining Grammars. In Dowty, D., Karttunen, L., and Zwicky, A., editors, *Natural Language Processing – Theoretical, Computational, and Psychological Perspectives*. Cambridge.

Joshi, A. K., Levy, L. S., and Takahashi, M. (1975). Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1).

Kobele, G. M. (2006). *Generating Copies: An investigation into Structural Identity in Language and Grammar*. PhD thesis, UCLA.

Konieczny, L. (2000). Locality and parsing complexity. *Journal of Psycholinguistic Research*, 29(6):627–645.

Kruijff, G.-J. M. and Vasishth, S. (2003). Quantifying word order freedom in natural language: Implications for sentence processing. Proceedings of the Architectures and Mechanisms for Language Processing conference.

Kuhlmann, M. (2007). *Dependency Structures and Lexicalized Grammars*. PhD thesis, Saarland University.

Kuhlmann, M. and Möhl, M. (2007). Mildly context-sensitive dependency languages. In *Proceedings of ACL*.

Kuhlmann, M. and Nivre, J. (2006). Mildly non-projective dependency structures. In *Proceedings of COLING/ACL*.

Levy, R. and Manning, C. (2004). Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation. In *Proceedings of ACL*.

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

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. (2005b). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of ACL*.

Miller, P. (2000). *Strong Generative Capacity: The Semantics of Linguistic Formalism*. Cambridge.

Pollard, C. (1984). *Generalized Phrase Structure Grammars, Head Grammars, and Natural Languages*. PhD thesis, Stanford.

Vijay-Shanker, K., Weir, D. J., and Joshi, A. K. (1987). Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of ACL*.

Positive Results for Parsing with a Bounded Stack using a Model-Based Right-Corner Transform

William Schuler
Dept. of Computer Science and Engineering
Minneapolis, MN
schuler@cs.umn.edu

Abstract

Statistical parsing models have recently been proposed that employ a bounded stack in time-series (left-to-right) recognition, using a right-corner transform defined over training trees to minimize stack use (Schuler et al., 2008). Corpus results have shown that a vast majority of naturally-occurring sentences can be parsed in this way using a very small stack bound of three to four elements. This suggests that the standard cubic-time CKY chart-parsing algorithm, which implicitly assumes an *unbounded* stack, may be wasting probability mass on trees whose complexity is beyond human recognition or generation capacity. This paper first describes a version of the right-corner transform that is defined over entire probabilistic grammars (cast as infinite sets of generable trees), in order to ensure a fair comparison between bounded-stack and unbounded PCFG parsing using a common underlying model; then it presents experimental results that show a bounded-stack right-corner parser using a transformed version of a grammar significantly outperforms an unbounded-stack CKY parser using the original grammar.

1 Introduction

Statistical parsing models have recently been proposed that employ a bounded stack in time-series (left-to-right) recognition, in order to directly and tractably incorporate incremental phenomena such as (co-)reference or disfluency into parsing decisions (Schuler et al., 2008; Miller and Schuler, 2008). These models make use of a right-corner tree transform, based on the left-corner transform described by Johnson (1998), and are supported by corpus results suggesting that most sentences (in English, at least) can be parsed using a very small stack bound of three to four elements (Schuler et al., 2008). This raises an interesting question: if most sentences can be recognized with only three or four elements of stack memory, is the standard cubic-time CKY chart-parsing algorithm, which implicitly assumes an *unbounded* stack, wasting probability mass on trees whose complexity is beyond human recognition or generation capacity?

This paper presents parsing accuracy results using transformed and untransformed versions of a corpus-trained probabilistic context-free grammar suggesting that this is indeed the case. Experimental results show a bounded-memory time-series parser using a transformed version of a grammar significantly outperforms an unbounded-stack CKY parser using the original grammar.

Unlike the tree-based transforms described previously, the model-based transform described in this paper does not introduce additional context from corpus data beyond that contained in the original probabilistic grammar, making it possible to present a fair comparison between bounded- and unbounded-stack versions of the same model. Since this transform takes a probabilistic grammar as input, it can also easily accommodate horizontal and vertical Markovisation (annotating grammar symbols with parent and sibling categories) as described by Collins (1997) and subsequently.

The remainder of this paper is organized as follows: Section 2 describes related approaches to parsing with stack bounds; Section 3 describes an existing bounded-stack parsing framework using a right-corner transform defined over individual trees; Section 4 describes a redefinition of this transform to ap-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 344–352,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

ply to entire probabilistic grammars, cast as infinite sets of generable trees; and Section 5 describes an evaluation of this transform on the Wall Street Journal corpus of the Penn Treebank showing improved results for a transformed bounded-stack version of a probabilistic grammar over the original unbounded grammar.

2 Related Work

The model examined here is formally similar to Combinatorial Categorial Grammar (CCG) (Steedman, 2000). But the CCG account is a competence model as well as a performance model, in that it seeks to unify category representations used in processing with learned generalizations about argument structure; whereas the model described in this paper is exclusively a performance model, allowing generalizations about lexical argument structures to be learned in some other representation, then combined with probabilistic information about parsing strategies to yield a set of derived incomplete constituents. As a result, the model described in this paper has a freer hand to satisfy strict working memory bounds, which may not permit some of the alternative composition operations proposed in the CCG account, thought to be associated with available prosody and quantifier scope analyses.[1]

Other models (Abney and Johnson, 1991; Gibson, 1991) seek to explain human processing difficulties as a result of memory capacity limits in parsing ordinary phrase structure trees. The Abney-Johnson and Gibson models adopt a left-corner parsing strategy, of which the right-corner transform described in this paper is a variant, in order to minimize memory usage. But the transform-based model described in this paper exploits a conception of chunking (Miller, 1956) — in this case, grouping recognized words into stacked-up incomplete constituents — to operate within much stricter estimates of human short-term memory bounds (Cowan, 2001) than assumed by Abney and Johnson.

[1]The lack of support for some of these available scope analyses may not necessarily be problematic for the present model. The complexity of interpreting nested raised quantifiers may place them beyond the capability of human interactive incremental interpretation, but not beyond the capability of post-hoc interpretation (understood after the listener has had time to think about it).

Several existing incremental systems are organized around a left-corner parsing strategy (Roark, 2001; Henderson, 2004). But these systems generally keep large numbers of constituents open for modifier attachment in each hypothesis. This allows modifiers to be attached as right children of any such open constituent. But if any number of open constituents are allowed, then either the assumption that stored elements have fixed syntactic (and semantic) structure will be violated, or the assumption that syntax operates within a bounded memory store will be violated, both of which are psycholinguistically attractive as simplifying assumptions. The HHMM model examined in this paper upholds both the fixed-element and bounded-memory assumptions by hypothesizing fixed reductions of right child constituents into incomplete parents in the same memory element, to make room for new constituents that may be introduced at a later time. These in-element reductions are defined naturally on phrase structure trees as the result of aligning right-corner transformed constituent structures to sequences of random variables in a factored time-series model.

3 Background

The recognition model examined in this paper is a factored time-series model, based on a Hierarchic Hidden Markov Model (Murphy and Paskin, 2001), which probabilistically estimates the contents of a memory store of three to four partially-completed constituents over time. Probabilities for expansions, transitions and reductions in this model can be defined over trees in a training corpus, transformed and mapped to the random variables in an HHMM (Schuler et al., 2008). In Section 4 these probabilities will be computed directly from a probabilistic context-free grammar, in order to evaluate the contribution of stack bounds without introducing additional corpus context into the model.

3.1 A Bounded-Stack Model

HHMMs are factored HMMs which mimic a bounded-memory pushdown automaton (PDA), supporting simple push and pop operations on a bounded stack-like memory store.

HMMs characterize speech or text as a sequence

of hidden states q_t (in this case, stacked-up syntactic categories) and observed states o_t (in this case, words) at corresponding time steps t. A most likely sequence of hidden states $\hat{q}_{1..T}$ can then be hypothesized given any sequence of observed states $o_{1..T}$:

$$\hat{q}_{1..T} = \underset{q_{1..T}}{\operatorname{argmax}} \mathsf{P}(q_{1..T} \mid o_{1..T}) \quad (1)$$

$$= \underset{q_{1..T}}{\operatorname{argmax}} \mathsf{P}(q_{1..T}) \cdot \mathsf{P}(o_{1..T} \mid q_{1..T}) \quad (2)$$

$$\stackrel{\text{def}}{=} \underset{q_{1..T}}{\operatorname{argmax}} \prod_{t=1}^{T} \mathsf{P}_{\Theta_{\text{A}}}(q_t \mid q_{t-1}) \cdot \mathsf{P}_{\Theta_{\text{B}}}(o_t \mid q_t) \quad (3)$$

using Bayes' Law (Equation 2) and Markov independence assumptions (Equation 3) to define a full $\mathsf{P}(q_{1..T} \mid o_{1..T})$ probability as the product of a *Transition Model (Θ_{A})* prior probability $\mathsf{P}(q_{1..T}) \stackrel{\text{def}}{=} \prod_t \mathsf{P}_{\Theta_{\text{A}}}(q_t \mid q_{t-1})$ and an *Observation Model (Θ_{B})* likelihood probability $\mathsf{P}(o_{1..T} \mid q_{1..T}) \stackrel{\text{def}}{=} \prod_t \mathsf{P}_{\Theta_{\text{B}}}(o_t \mid q_t)$.

Transition probabilities $\mathsf{P}_{\Theta_{\text{A}}}(q_t \mid q_{t-1})$ over complex hidden states q_t can be modeled using synchronized levels of stacked-up component HMMs in an HHMM. HHMM transition probabilities are calculated in two phases: a *reduce* phase (resulting in an intermediate, marginalized state f_t), in which component HMMs may terminate; and a *shift* phase (resulting in a modeled state q_t), in which unterminated HMMs transition, and terminated HMMs are reinitialized from their parent HMMs. Variables over intermediate f_t and modeled q_t states are factored into sequences of depth-specific variables – one for each of D levels in the HHMM hierarchy:

$$f_t = \langle f_t^1 \ldots f_t^D \rangle \quad (4)$$

$$q_t = \langle q_t^1 \ldots q_t^D \rangle \quad (5)$$

Transition probabilities are then calculated as a product of transition probabilities at each level, using level-specific *reduce* $\Theta_{\text{R},d}$ and *shift* $\Theta_{\text{S},d}$ models:

$$\mathsf{P}_{\Theta_{\text{A}}}(q_t | q_{t-1}) = \sum_{f_t} \mathsf{P}(f_t | q_{t-1}) \cdot \mathsf{P}(q_t | f_t\, q_{t-1}) \quad (6)$$

$$\stackrel{\text{def}}{=} \sum_{f_t^{1..D}} \prod_{d=1}^{D} \mathsf{P}_{\Theta_{\text{R},d}}(f_t^d | f_t^{d+1} q_{t-1}^d q_{t-1}^{d-1}) \cdot \mathsf{P}_{\Theta_{\text{S},d}}(q_t^d | f_t^{d+1} f_t^d\, q_{t-1}^d q_t^{d-1}) \quad (7)$$

with f_t^{D+1} and q_t^0 defined as constants. In Viterbi decoding, the sums are replaced with argmax operators. This decoding process preserves ambiguity by maintaining competing analyses of the entire memory store. A graphical representation of an HHMM with three levels is shown in Figure 1.

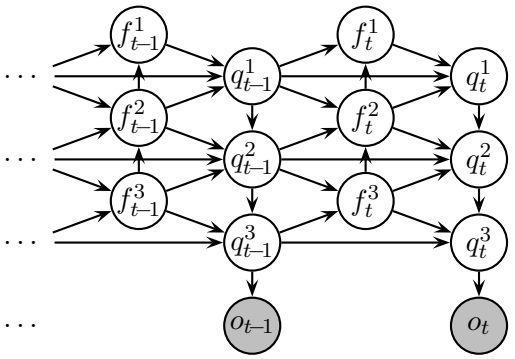


Figure 1: Graphical representation of a Hierarchic Hidden Markov Model. Circles denote random variables, and edges denote conditional dependencies. Shaded circles are observations.

Shift and reduce probabilities can then be defined in terms of finitely recursive Finite State Automata (FSAs) with probability distributions over transition, recursive expansion, and final-state status of states at each hierarchy level. In the version of HHMMs used in this paper, each intermediate variable is a reduction or non-reduction state $f_t^d \in G \cup \{\mathbf{1}, \mathbf{0}\}$ (indicating, respectively, a complete reduced constituent of some grammatical category from domain G, or a failure to reduce due to an 'active' transition being performed, or a failure to reduce due to an 'awaited' transition being performed, as defined in Section 4.3); and each modeled variable is a syntactic state $q_t^d \in G \times G$ (describing an incomplete constituent consisting of an active grammatical category from domain G and an awaited grammatical category from domain G). An intermediate variable f_t^d at depth d may indicate reduction or non-reduction according to $\Theta_{\text{F-Rd},d}$ if there is a reduction at the depth level immediately below d, but must indicate non-reduction (**0**) with probability 1 if there was no reduction below:[2]

$$\mathsf{P}_{\Theta_{\text{R},d}}(f_t^d \mid f_t^{d+1} q_{t-1}^d q_{t-1}^{d-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } f_t^{d+1} \notin G : [f_t^d = \mathbf{0}] \\ \text{if } f_t^{d+1} \in G : \mathsf{P}_{\Theta_{\text{F-Rd},d}}(f_t^d \mid q_{t-1}^d, q_{t-1}^{d-1}) \end{cases} \quad (8)$$

[2] Here $[\cdot]$ is an indicator function: $[\phi] = 1$ if ϕ is true, 0 otherwise.

where $f_t^{D+1} \in G$ and $q_t^0 = \mathbf{ROOT}$.

Shift probabilities over the modeled variable q_t^d at each level are defined using level-specific transition $\Theta_{\text{Q-Tr},d}$ and expansion $\Theta_{\text{Q-Ex},d}$ models:

$$\mathsf{P}_{\Theta_{\text{S},d}}(q_t^d \mid f_t^{d+1}\, f_t^d\, q_{t-1}^d\, q_t^{d-1}) \stackrel{\text{def}}{=}$$
$$\begin{cases} \text{if } f_t^{d+1} \notin G,\ f_t^d \notin G : [q_t^d = q_{t-1}^d] \\ \text{if } f_t^{d+1} \in G,\ f_t^d \notin G : \mathsf{P}_{\Theta_{\text{Q-Tr},d}}(q_t^d \mid f_t^{d+1}\, f_t^d\, q_{t-1}^d\, q_t^{d-1}) \\ \text{if } f_t^{d+1} \in G,\ f_t^d \in G : \mathsf{P}_{\Theta_{\text{Q-Ex},d}}(q_t^d \mid q_t^{d-1}) \end{cases} \tag{9}$$

where $f_t^{D+1} \in G$ and $q_t^0 = \mathbf{ROOT}$. This model is conditioned on reduce variables at and immediately below the current FSA level. If there is no reduction immediately below the current level (the first case above), it deterministically copies the current FSA state forward to the next time step. If there is a reduction immediately below the current level but no reduction at the current level (the second case above), it transitions the FSA state at the current level, according to the distribution $\Theta_{\text{Q-Tr},d}$. And if there is a reduction at the current level (the third case above), it re-initializes this state given the state at the level above, according to the distribution $\Theta_{\text{Q-Ex},d}$. The overall effect is that higher-level FSAs are allowed to transition only when lower-level FSAs terminate. An HHMM therefore behaves like a probabilistic implementation of a pushdown automaton (or shift–reduce parser) with a finite stack, where the maximum stack depth is equal to the number of levels in the HHMM hierarchy.

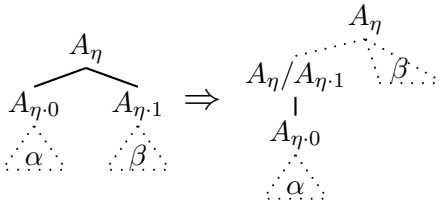
3.2 Tree-Based Transforms

The right-corner transform used in this paper is simply the left-right dual of a left-corner transform (Johnson, 1998). It transforms all right branching sequences in a phrase structure tree into left branching sequences of symbols of the form $A_\eta/A_{\eta\cdot\mu}$, denoting an incomplete instance of an 'active' category A_η lacking an instance of an 'awaited' category $A_{\eta\cdot\mu}$ yet to come.[3] These incomplete constituent categories have the same form and much of the same meaning as non-constituent categories in a Combinatorial Categorial Grammar (Steedman, 2000).

[3]Here η and μ are node addresses in a binary-branching tree, defined as paths of left (0) or right (1) branches from the root.

Rewrite rules for the right-corner transform are shown below:[4]

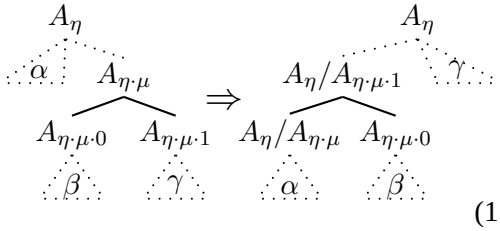
- Beginning case: the top of a right-expanding sequence in an ordinary phrase structure tree is mapped to the bottom of a left-expanding sequence in a right-corner transformed tree:



(10)

  This case of the right-corner transform may be considered a constrained version of CCG type raising.

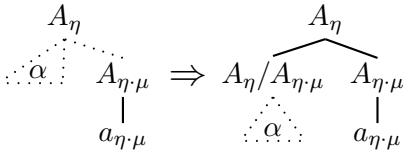
- Middle case: each subsequent branch in a right-expanding sequence of an ordinary phrase structure tree is mapped to a branch in a left-expanding sequence of the transformed tree:



(11)

  This case of the right-corner transform may be considered a constrained version of CCG forward function composition.

- Ending case: the bottom of a right-expanding sequence in an ordinary phrase structure tree is mapped to the top of a left-expanding sequence in a right-corner transformed tree:

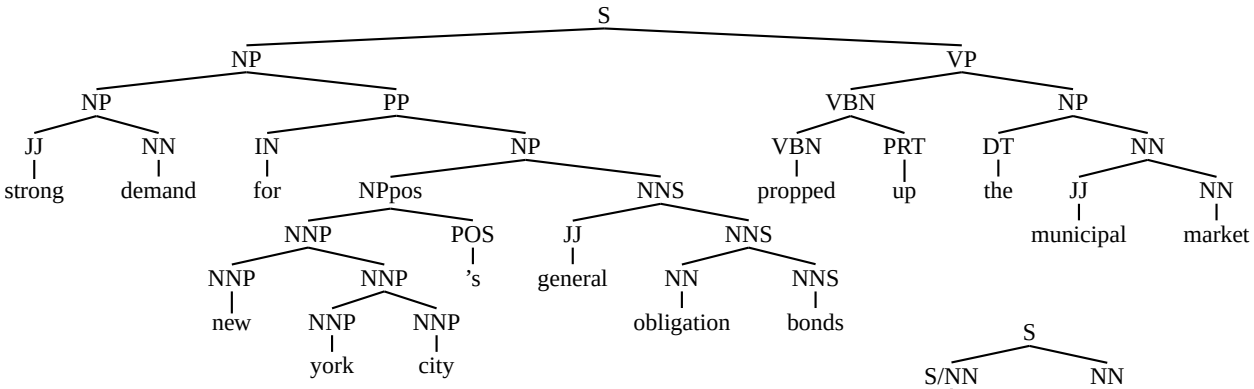


(12)

  This case of the right-corner transform may be considered a constrained version of CCG forward function application.

[4]These rules can be applied recursively from bottom up on a source tree, synchronously associating subtree structures matched to variables α, β, and γ on the left side of each rule with transformed representations of these subtree structures on the right.

a) binary-branching phrase structure tree:



b) result of right-corner transform:

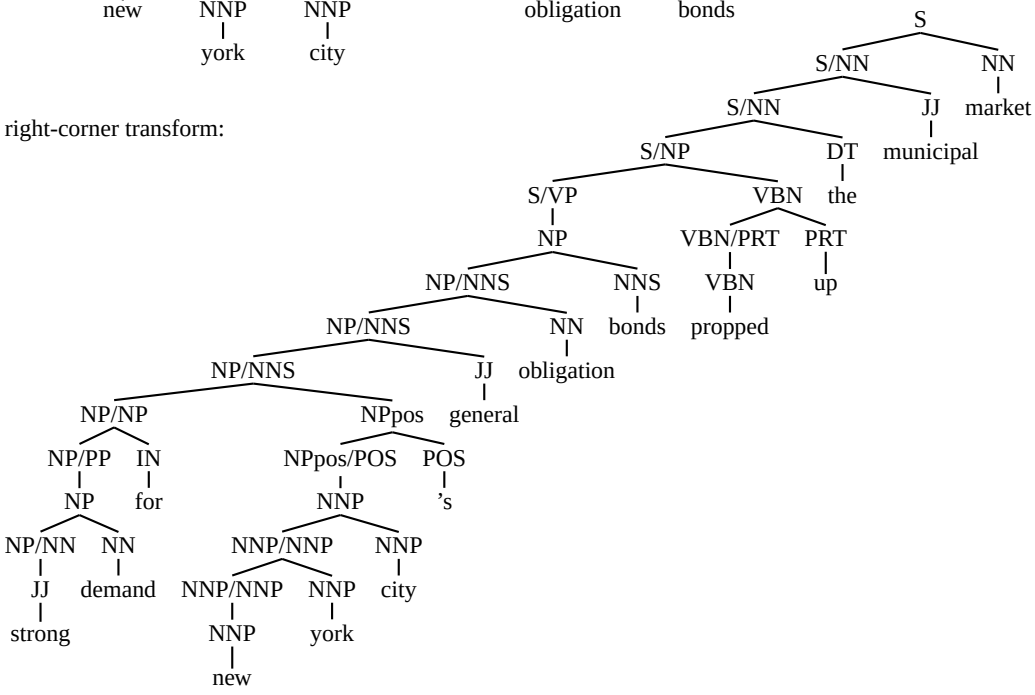


Figure 2: Trees resulting from a) a sample phrase structure tree for the sentence *Strong demand for New York City's general obligations bonds propped up the municipal market*, and b) a right-corner transform of this tree. Sequences of left children are recognized from the bottom up through in-element transitions in a Hierarchic Hidden Markov Model. Right children are recognized by expanding to additional stack elements.

The completeness of the above transform rules can be demonstrated by the fact that they cover all possible subtree configurations (with the exception of bare terminals, which are simply copied). The soundness of the above transform rules can be demonstrated by the fact that each rule transforms a right-branching subtree into a left-branching subtree labeled with an incomplete constituent.

An example of a right-corner transformed tree is shown in Figure 2(b). An important property of this transform is that it is reversible. Rewrite rules for reversing a right-corner transform are simply the converse of those shown above.

Sequences of left children in the resulting mostly-left-branching trees are recognized from the bottom up, through transitions at the same stack element. Right children, which are much less frequent in the resulting trees, are recognized through cross-element expansions in a bounded-stack recognizer.

4 Model-Based Transforms

In order to compare bounded- and unbounded-stack versions of the same model, the formulation of the right-corner and bounded-stack transforms introduced in this paper does not map trees to trees, but rather maps probability models to probability

models. This eliminates complications in comparing models with different numbers of dependent variables — and thus different numbers of free parameters — because the model which ordinarily has more free parameters (the HHMM, in this case) is derived from the model that has fewer (the PCFG). Since they are derived from a simpler underlying model, the additional parameters of the HHMM are not free.

Mapping probability models from one format to another can be thought of as mapping the infinite sets of trees that are defined by these models from one format to another. Probabilities in the transformed model are therefore defined by calculating probabilities for the relevant substructures in the source model, then marginalizing out the values of nodes in these structures that do not appear in the desired expression in the target model.

A bounded-stack HHMM $\Theta_{\text{Q,F}}$ can therefore be derived from an unbounded PCFG Θ_{G} by:

1. organizing the rules in the source PCFG model Θ_{G} into direction-specific versions (distinguishing rules for expanding left and right children, which occur respectively as active and awaited constituent categories in incomplete constituent labels);

2. enforcing depth limits on these direction-specific rules; and

3. mapping these probabilities to HHMM random variable positions at the appropriate depth.

4.1 Direction-specific rules

An inspection of the tree-based right-corner transform rewrites defined in Section 3.2 will show two things: first, that constituents occurring as left children in an original tree (with addresses ending in '0') always become active constituents (occurring before the slash, or without a slash) in incomplete constituent categories, and constituents occurring as right children in an original tree (with addresses ending in '1') always become awaited constituents (occurring after the slash); and second, that left children expand locally downward in the transformed tree (so each $A_{\eta \cdot 0}/\ldots$ locally dominates $A_{\eta \cdot 0 \cdot 0}/\ldots$), whereas right children expand locally upward (so each $\ldots/A_{\eta \cdot 1}$ is locally dominated by $\ldots/A_{\eta \cdot 1 \cdot 1}$). This means that rules from the original grammar — if distinguished into rules applying only to left and right children (active and awaited constituents) — can still be locally modeled following a right-corner transform. A transformed tree can be generated in this way by expanding downward along the active constituents in a transformed tree, then turning around and expanding upward to fill in the awaited constituents, then turning around again to generate the active constituents at the next depth level, and so on.

4.2 Depth bounds

The locality of the original grammar rules in a right-corner transformed tree allows memory limits on incomplete constituents to be applied directly as depth bounds in the zig-zag generation traversal defined above. These depth limits correspond directly to the depth levels in an HHMM.

In the experiments described in Section 5, direction-specific and depth-specific versions of the original grammar rules are implemented in an ordinary CKY-style dynamic-programming parser, and can therefore simply be cut off at a particular depth level with no renormalization.

But in an HHMM, this will result in *label-bias* effects, in which expanded constituents may have no valid reduction, forcing the system to define distributions for composing constituents that are not compatible. For example, if a constituent is expanded at depth D, and that constituent has no expansions that can be completely processed within depth D, it will not be able to reduce, and will remain incompatible with the incomplete constituent above it. Probabilities for depth-bounded rules must therefore be renormalized to the domain of allowable trees that can be generated within D depth levels, in order to guarantee consistent probabilities for HHMM recognition.

This is done by determining the (depth- and direction-specific) probability $\mathsf{P}_{\Theta_{\text{B-L},d}}(\mathbf{1} \mid A_{\eta \cdot 0})$ or $\mathsf{P}_{\Theta_{\text{B-R},d}}(\mathbf{1} \mid A_{\eta \cdot 1})$ that a tree generated at each depth d and rooted by a left or right child will fit within depth D. These probabilities are then estimated using an approximate inference algorithm, similar to that used in value iteration (Bellman, 1957), which estimates probabilities of infinite trees by exploiting the fact that increasingly longer trees contribute exponentially decreasing probability mass (since each non-terminal expansion must

avoid generating a terminal with some probability at each step from the top down), so a sum over probabilities of trees with increasing length k is guaranteed to converge. The algorithm calculates probabilities of trees with increasing length k until convergence, or to some arbitrary limit K:

$$\mathsf{P}_{\Theta_{\text{B-L},d,k}}(\mathbf{1} \mid A_{\eta\cdot 0}) \stackrel{\text{def}}{=} \sum_{\substack{A_{\eta\cdot 1\cdot 0},\\ A_{\eta\cdot 1\cdot 1}}} \mathsf{P}_{\Theta_{\text{G}}}(A_{\eta\cdot 0} \rightarrow A_{\eta\cdot 0\cdot 0}\ A_{\eta\cdot 0\cdot 1}) \cdot \mathsf{P}_{\Theta_{\text{B-L},d,k-1}}(\mathbf{1} \mid A_{\eta\cdot 0\cdot 0}) \cdot \mathsf{P}_{\Theta_{\text{B-R},d,k-1}}(\mathbf{1} \mid A_{\eta\cdot 0\cdot 1}) \quad (13)$$

$$\mathsf{P}_{\Theta_{\text{B-R},d,k}}(\mathbf{1} \mid A_{\eta\cdot 1}) \stackrel{\text{def}}{=} \sum_{\substack{A_{\eta\cdot 1\cdot 0},\\ A_{\eta\cdot 1\cdot 1}}} \mathsf{P}_{\Theta_{\text{G}}}(A_{\eta\cdot 1} \rightarrow A_{\eta\cdot 1\cdot 0}\ A_{\eta\cdot 1\cdot 1}) \cdot \mathsf{P}_{\Theta_{\text{B-L},d+1,k-1}}(\mathbf{1} \mid A_{\eta\cdot 1\cdot 0}) \cdot \mathsf{P}_{\Theta_{\text{B-R},d,k-1}}(\mathbf{1} \mid A_{\eta\cdot 1\cdot 1}) \quad (14)$$

Normalized probability distributions for depth-bounded expansions $\Theta_{\text{G-L},d}$ and $\Theta_{\text{G-R},d}$ can now be calculated using converged $\Theta_{\text{B-L},d}$ and $\Theta_{\text{B-R},d}$ estimates:

$$\mathsf{P}_{\Theta_{\text{G-L},d}}(A_{\eta\cdot 0} \rightarrow A_{\eta\cdot 0\cdot 0}\ A_{\eta\cdot 0\cdot 1}) \stackrel{\text{def}}{=} \mathsf{P}_{\Theta_{\text{G}}}(A_{\eta\cdot 0} \rightarrow A_{\eta\cdot 0\cdot 0}\ A_{\eta\cdot 0\cdot 1}) \cdot \mathsf{P}_{\Theta_{\text{B-L},d}}(\mathbf{1} \mid A_{\eta\cdot 0\cdot 0}) \cdot \mathsf{P}_{\Theta_{\text{B-R},d}}(\mathbf{1} \mid A_{\eta\cdot 0\cdot 1}) \quad (15)$$

$$\mathsf{P}_{\Theta_{\text{G-R},d}}(A_{\eta\cdot 1} \rightarrow A_{\eta\cdot 1\cdot 0}\ A_{\eta\cdot 1\cdot 1}) \stackrel{\text{def}}{=} \mathsf{P}_{\Theta_{\text{G}}}(A_{\eta\cdot 1} \rightarrow A_{\eta\cdot 1\cdot 0}\ A_{\eta\cdot 1\cdot 1}) \cdot \mathsf{P}_{\Theta_{\text{B-L},d+1}}(\mathbf{1} \mid A_{\eta\cdot 1\cdot 0}) \cdot \mathsf{P}_{\Theta_{\text{B-R},d}}(\mathbf{1} \mid A_{\eta\cdot 1\cdot 1}) \quad (16)$$

4.3 HHMM probabilities

Converting PCFGs to HHMMs requires the calculation of expected frequencies $\mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_\eta \stackrel{*}{\rightarrow} A_{\eta\cdot\mu})$ of generating symbols $A_{\eta\cdot\mu}$ in the left-progeny of a nonterminal symbol A_η (in other words, of $A_{\eta\cdot\mu}$ being a left child of A_η, or a left child of a left child of A_η, etc.). This is done by summing over subtrees of increasing length k using the same approximate inference technique described in Section 4.2, which guarantees convergence since each subtree of increasing length contributes exponentially decreasing probability mass to the sum:

$$\mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_\eta \stackrel{*}{\rightarrow} A_{\eta\cdot\mu}) = \sum_{k=0}^{\infty} \mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_\eta \stackrel{k}{\rightarrow} A_{\eta\cdot\mu}) \quad (17)$$

where:

$$\mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_\eta \stackrel{k}{\rightarrow} A_{\eta\cdot 0^k}) = \sum_{\substack{A_{\eta\cdot 0^{k-1}},\\ A_{\eta\cdot 0^{k-1}\cdot 1}}} \mathsf{P}_{\Theta_{\text{G-L}^*,d}}(A_\eta \stackrel{k-1}{\rightarrow} A_{\eta\cdot 0^{k-1}}) \cdot \mathsf{P}_{\Theta_{\text{G-L},d}}(A_{\eta\cdot 0^{k-1}} \rightarrow A_{\eta\cdot 0^k}\ A_{\eta\cdot 0^{k-1}\cdot 1}) \quad (18)$$

and $\mathsf{P}_{\Theta_{\text{G-L}^*,d}}(A_\eta \stackrel{0}{\rightarrow} A'_\eta) = [A_\eta = A'_\eta]$.

A complete HHMM can now be defined using depth-bounded right-corner PCFG probabilities. HHMM probabilities will be defined over syntactic states consisting of incomplete constituent categories $A_\eta/A_{\eta\cdot\mu}$.

Expansions depend on only the incomplete constituent category $../A_\eta$ (for any active category '..') at q_t^{d-1}:

$$\mathsf{P}_{\Theta_{\text{Q-Ex},d}}(a_{\eta\cdot 0\cdot\mu} \mid ../A_\eta) = \frac{\sum_{A_{\eta\cdot 0},\ A_{\eta\cdot 1}} \mathsf{P}_{\Theta_{\text{G-R},d-1}}(A_\eta \rightarrow A_{\eta\cdot 0}\ A_{\eta\cdot 1}) \cdot \mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_{\eta\cdot 0} \stackrel{*}{\rightarrow} a_{\eta\cdot 0\cdot\mu})}{\sum_{\substack{A_{\eta\cdot 0},\\ a_{\eta\cdot 0\cdot\mu}}} \sum_{A_{\eta\cdot 1}} \mathsf{P}_{\Theta_{\text{G-R},d-1}}(A_\eta \rightarrow A_{\eta\cdot 0}\ A_{\eta\cdot 1}) \cdot \mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_{\eta\cdot 0} \stackrel{*}{\rightarrow} a_{\eta\cdot 0\cdot\mu})} \quad (19)$$

Transitions depend on whether an 'active' or 'awaited' transition was performed at the current level. If an active transition was performed (where $f_t^d = \mathbf{1}$), the transition depends on only the incomplete constituent category $A_{\eta\cdot 0\cdot\mu\cdot 0}/..$ (for any awaited category '..') at q_{t-1}^d, and the incomplete constituent category $../A_\eta$ (for any active category '..') at q_{t-1}^{d-1}:

$$\mathsf{P}_{\Theta_{\text{Q-Tr},d}}(A_{\eta\cdot 0\cdot\mu}/A_{\eta\cdot 0\cdot\mu\cdot 1} \mid \mathbf{1},\ A_{\eta\cdot 0\cdot\mu\cdot 0}/.., ../A_\eta) = \frac{\sum_{\substack{A_{\eta\cdot 0},\\ A_{\eta\cdot 1}}} \mathsf{P}_{\Theta_{\text{G-R},d-1}}(A_\eta \rightarrow A_{\eta\cdot 0}\ A_{\eta\cdot 1}) \cdot \frac{\mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_{\eta\cdot 0} \stackrel{*}{\rightarrow} A_{\eta\cdot 0\cdot\mu})}{\mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_{\eta 0} \stackrel{*}{\rightarrow} A_{\eta 0\mu 0}) - \mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_{\eta 0} \stackrel{0}{\rightarrow} A_{\eta 0\mu 0})} \cdot \mathsf{P}_{\Theta_{\text{G-L},d}}(A_{\eta\cdot 0\cdot\mu} \rightarrow A_{\eta\cdot 0\cdot\mu\cdot 0}\ A_{\eta\cdot 0\cdot\mu\cdot 1})}{\sum_{\substack{A_{\eta\cdot 0},\\ A_{\eta\cdot 1},\\ A_{\eta\cdot 0\cdot\mu},\\ A_{\eta\cdot 0\cdot\mu\cdot 1}}} \mathsf{P}_{\Theta_{\text{G-R},d-1}}(A_\eta \rightarrow A_{\eta\cdot 0}\ A_{\eta\cdot 1}) \cdot \frac{\mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_{\eta\cdot 0} \stackrel{*}{\rightarrow} A_{\eta\cdot 0\cdot\mu})}{\mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_{\eta 0} \stackrel{*}{\rightarrow} A_{\eta 0\mu 0}) - \mathsf{F}_{\Theta_{\text{G-L}^*,d}}(A_{\eta 0} \stackrel{0}{\rightarrow} A_{\eta 0\mu 0})} \cdot \mathsf{P}_{\Theta_{\text{G-L},d}}(A_{\eta\cdot 0\cdot\mu} \rightarrow A_{\eta\cdot 0\cdot\mu\cdot 0}\ A_{\eta\cdot 0\cdot\mu\cdot 1})} \quad (20)$$

If an awaited transition was performed (where $f_t^d = \mathbf{0}$), the transition depends on only the complete constituent category $A_{\eta\cdot\mu\cdot 0}$ at f_t^{d+1}, and the incomplete

constituent category $A_\eta / A_{\eta\cdot\mu}$ at q_{t-1}^d:

$$\mathsf{P}_{\Theta_{\text{Q-Tr},d}}(A_\eta / A_{\eta\cdot\mu\cdot 1} \mid \mathbf{0},\ A_{\eta\cdot\mu\cdot 0},\ A_\eta / A_{\eta\cdot\mu}) = \frac{\mathsf{P}_{\Theta_{\text{G-R},d}}(A_{\eta\cdot\mu} \rightarrow A_{\eta\cdot\mu\cdot 0}\ A_{\eta\cdot\mu\cdot 1})}{\sum_{A_{\eta\cdot\mu\cdot 1}} \mathsf{P}_{\Theta_{\text{G-R},d}}(A_{\eta\cdot\mu} \rightarrow A_{\eta\cdot\mu\cdot 0}\ A_{\eta\cdot\mu\cdot 1})} \quad (21)$$

Reduce probabilities depend on the complete constituent category at f_t^{d+1}, and the incomplete constituent category $A_{\eta\cdot 0\cdot\mu\cdot 0}/..$ (for any awaited category '..') at q_{t-1}^d, and the incomplete constituent category $../A_\eta$ (for any active category '..') at q_{t-1}^{d-1}. If the complete constituent category at f_t^{d+1} does not match the awaited category of q_{t-1}^d, the probability is $[f_t^d = f 0]$. If the complete constituent category at f_t^{d+1} does match the awaited category of q_{t-1}^d:

$$\mathsf{P}_{\Theta_{\text{F-Rd},d}}(\mathbf{1} \mid A_{\eta\cdot 0\cdot\mu}/..,\ ../A_\eta) = \frac{\sum_{A_{\eta\cdot 0}, A_{\eta\cdot 1}} \mathsf{P}_{\Theta_{\text{G-R},d-1}}(A_\eta \rightarrow A_{\eta\cdot 0}\ A_{\eta\cdot 1}) \cdot \begin{pmatrix} \mathsf{F}_{\Theta_{\text{G-L*},d}}(A_{\eta\cdot 0} \overset{*}{\rightarrow} A_{\eta\cdot 0\cdot\mu}) \\ -\mathsf{F}_{\Theta_{\text{G-L*},d}}(A_{\eta\cdot 0} \overset{0}{\rightarrow} A_{\eta\cdot 0\cdot\mu}) \end{pmatrix}}{\sum_{A_{\eta\cdot 0}, A_{\eta\cdot 1}} \mathsf{P}_{\Theta_{\text{G-R},d-1}}(A_\eta \rightarrow A_{\eta\cdot 0}\ A_{\eta\cdot 1}) \cdot \mathsf{F}_{\Theta_{\text{G-L*},d}}(A_{\eta\cdot 0} \overset{*}{\rightarrow} A_{\eta\cdot 0\cdot\mu})} \quad (22)$$

and:

$$\mathsf{P}_{\Theta_{\text{F-Rd},d}}(A_{\eta\cdot 0\cdot\mu} \mid A_{\eta\cdot 0\cdot\mu}/..,\ ../A_\eta) = \frac{\sum_{A_{\eta\cdot 0}, A_{\eta\cdot 1}} \mathsf{P}_{\Theta_{\text{G-R},d-1}}(A_\eta \rightarrow A_{\eta\cdot 0}\ A_{\eta\cdot 1}) \cdot \mathsf{F}_{\Theta_{\text{G-L*},d}}(A_{\eta\cdot 0} \overset{0}{\rightarrow} A_{\eta\cdot 0\cdot\mu})}{\sum_{A_{\eta\cdot 0}, A_{\eta\cdot 1}} \mathsf{P}_{\Theta_{\text{G-R},d-1}}(A_\eta \rightarrow A_{\eta\cdot 0}\ A_{\eta\cdot 1}) \cdot \mathsf{F}_{\Theta_{\text{G-L*},d}}(A_{\eta\cdot 0} \overset{*}{\rightarrow} A_{\eta\cdot 0\cdot\mu})} \quad (23)$$

The correctness of the above distributions can be demonstrated by the fact that all terms other than $\Theta_{\text{G-L},d}$ and $\Theta_{\text{G-R},d}$ probabilities will cancel out in any sequence of transitions between an expansion and a reduction, leaving only those terms that would appear as factors in an ordinary PCFG parse.[5]

5 Results

A PCFG model was extracted from sections 2–21 of the Wall Street Journal Treebank. In order to keep the transform process manageable, punctuation was removed from the corpus, and rules occurring less frequently than 10 times in the corpus were deleted from the PCFG. The right-corner and bounded-stack transforms described in the previous section were then applied to the PCFG. The original and bounded PCFG models were evaluated in a CKY recognizer on sections 22–24 of the Treebank, with results shown in Table 1.[6] Results were significant only for sentences longer than 40 words. On these sentences, the bounded PCFG model achieves about a .15% reduction of error over the original PCFG ($p < .1$ using one-tailed pairwise t-test). This suggests that on long sentences the probability mass wasted due to parsing with an unbounded stack is substantial enough to impact parsing accuracy.

model (sect 22–24, len>40)	F
unbounded PCFG	66.03
bounded PCFG (D=4)	66.08

Table 1: Results of CKY parsing using bounded and unbounded PCFG.

6 Conclusion

Previous work has explored bounded-stack parsing using a right-corner transform defined on trees to minimize stack usage. HHMM parsers trained on applications of this tree-based transform of training corpora have shown improvements over ordinary PCFG models, but this may have been attributable to the richer dependencies of the HHMM.

This paper has presented an approximate inference algorithm for transforming entire PCFGs, rather than individual trees, into equivalent right-corner bounded-stack HHMMs. Moreover, a comparison with an untransformed PCFG model suggests that the probability mass wasted due to parsing with an unbounded stack is substantial enough to impact parsing accuracy.

Acknowledgments

This research was supported by NSF CAREER award 0447685 and by NASA under award NNX08AC36A. The views expressed are not necessarily endorsed by the sponsors.

[5] It is important to note, however, that these probabilities are not necessarily incrementally balanced, so this correctness only applies to parsing with an infinite beam.

[6] A CKY recognizer was used in both cases in order to avoid introducing errors due to model approximation or beam limits necessary for incremental processing with large grammars.

References

Steven P. Abney and Mark Johnson. 1991. Memory requirements and local ambiguities of parsing strategies. *J. Psycholinguistic Research*, 20(3):233–250.

Richard Bellman. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL '97)*.

Nelson Cowan. 2001. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24:87–185.

Edward Gibson. 1991. *A computational theory of human linguistic processing: Memory limitations and processing breakdown*. Ph.D. thesis, Carnegie Mellon.

James Henderson. 2004. Lookahead in deterministic left-corner parsing. In *Proc. Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, Barcelona, Spain.

Mark Johnson. 1998. Finite state approximation of constraint-based grammars using left-corner grammar transforms. In *Proceedings of COLING/ACL*, pages 619–623.

Tim Miller and William Schuler. 2008. A syntactic time-series model for parsing fluent and disfluent speech. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08)*.

George A. Miller. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97.

Kevin P. Murphy and Mark A. Paskin. 2001. Linear time inference in hierarchical HMMs. In *Proc. NIPS*, pages 833–840.

Brian Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2008. Toward a psycholinguistically-motivated model of language. In *Proceedings of COLING*, Manchester, UK, August.

Mark Steedman. 2000. *The syntactic process*. MIT Press/Bradford Books, Cambridge, MA.

Hierarchical Text Segmentation from Multi-Scale Lexical Cohesion

Jacob Eisenstein
Beckman Institute for Advanced Science and Technology
University of Illinois
Urbana, IL 61801
jacobe@illinois.edu

Abstract

This paper presents a novel unsupervised method for hierarchical topic segmentation. Lexical cohesion – the workhorse of unsupervised linear segmentation – is treated as a multi-scale phenomenon, and formalized in a Bayesian setting. Each word token is modeled as a draw from a pyramid of latent topic models, where the structure of the pyramid is constrained to induce a hierarchical segmentation. Inference takes the form of a coordinate-ascent algorithm, iterating between two steps: a novel dynamic program for obtaining the globally-optimal hierarchical segmentation, and collapsed variational Bayesian inference over the hidden variables. The resulting system is fast and accurate, and compares well against heuristic alternatives.

1 Introduction

Recovering structural organization from unformatted texts or transcripts is a fundamental problem in natural language processing, with applications to classroom lectures, meeting transcripts, and chatroom logs. In the unsupervised setting, a variety of successful systems have leveraged *lexical cohesion* (Halliday and Hasan, 1976) – the idea that topically-coherent segments display consistent lexical distributions (Hearst, 1994; Utiyama and Isahara, 2001; Eisenstein and Barzilay, 2008). However, such systems almost invariably focus on *linear* segmentation, while it is widely believed that discourse displays a hierarchical structure (Grosz and Sidner, 1986). This paper introduces the concept of *multi-scale lexical cohesion*, and leverages this idea in a Bayesian generative model for hierarchical topic segmentation.

The idea of multi-scale cohesion is illustrated by the following two examples, drawn from the Wikipedia entry for the city of Buenos Aires.

> There are over 150 city **bus** *lines* called **Colectivos** ... **Colectivos** in Buenos Aires do not have a fixed *timetable*, but *run* from 4 to several per *hour*, depending on the **bus** *line* and *time* of the day.
>
> The Buenos Aires **metro** has six *lines*, 74 **stations**, and 52.3 km of **track.** An expansion program is underway to extend existing *lines* into the outer *neighborhoods*. **Track** length is expected to reach 89 km...

The two sections are both part of a high-level segment on transportation. Words in bold are characteristic of the subsections (buses and trains, respectively), and do not occur elsewhere in the transportation section; words in italics occur throughout the high-level section, but not elsewhere in the article. This paper shows how multi-scale cohesion can be captured in a Bayesian generative model and exploited for unsupervised hierarchical topic segmentation.

Latent topic models (Blei et al., 2003) provide a powerful statistical apparatus with which to study discourse structure. A consistent theme is the treatment of individual words as draws from multinomial language models indexed by a hidden "topic" associated with the word. In latent Dirichlet allocation (LDA) and related models, the hidden topic for each word is unconstrained and unrelated to the hidden topic of neighboring words (given the parameters). In this paper, the latent topics are constrained to produce a hierarchical segmentation structure, as shown in Figure 1.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 353–361,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

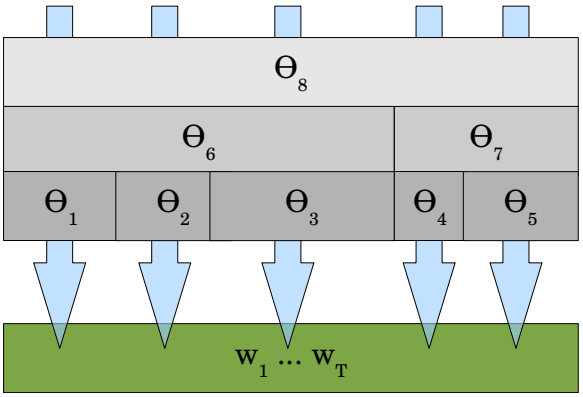


Figure 1: Each word w_t is drawn from a mixture of the language models located above t in the pyramid.

These structural requirements simplify inference, allowing the language models to be analytically marginalized. The remaining hidden variables are the scale-level assignments for each word token. Given marginal distributions over these variables, it is possible to search the entire space of hierarchical segmentations in polynomial time, using a novel dynamic program. Collapsed variational Bayesian inference is then used to update the marginals. This approach achieves high quality segmentation on multiple levels of the topic hierarchy.

Source code is available at `http://people.csail.mit.edu/jacobe/naacl09.html`.

2 Related Work

The use of lexical cohesion (Halliday and Hasan, 1976) in unsupervised topic segmentation dates back to Hearst's seminal TEXTTILING system (1994). Lexical cohesion was placed in a probabilistic (though not Bayesian) framework by Utiyama and Isahara (2001). The application of Bayesian topic models to text segmentation was investigated first by Blei and Moreno (2001) and later by Purver et al. (2006), using HMM-like graphical models for linear segmentation. Eisenstein and Barzilay (2008) extend this work by marginalizing the language models using the Dirichlet compound multinomial distribution; this permits efficient inference to be performed directly in the space of segmentations. All of these papers consider only linear topic segmentation; we introduce multi-scale lexical cohesion, which posits that the distribution of some words changes slowly with high-level topics, while others change rapidly with lower-level subtopics. This gives a principled mechanism to model hierarchical topic segmentation.

The literature on hierarchical topic segmentation is relatively sparse. Hsueh et al. (2006) describe a supervised approach that trains separate classifiers for topic and sub-topic segmentation; more relevant for the current work is the unsupervised method of Yaari (1997). As in TEXTTILING, cohesion is measured using cosine similarity, and agglomerative clustering is used to induce a dendrogram over paragraphs; the dendrogram is transformed into a hierarchical segmentation using a heuristic algorithm. Such heuristic approaches are typically brittle, as they include a number of parameters that must be hand-tuned. These problems can be avoided by working in a Bayesian probabilistic framework.

We note two orthogonal but related approaches to extracting nonlinear discourse structures from text. Rhetorical structure theory posits a hierarchical structure of discourse relations between spans of text (Mann and Thompson, 1988). This structure is richer than hierarchical topic segmentation, and the base level of analysis is typically more fine-grained – at the level of individual clauses. Unsupervised approaches based purely on cohesion are unlikely to succeed at this level of granularity.

Elsner and Charniak (2008) propose the task of conversation disentanglement from internet chatroom logs. Unlike hierarchical topic segmentation, conversational threads may be disjoint, with unrelated threads interposed between two utterances from the same thread. Elsner and Charniak present a supervised approach to this problem, but the development of cohesion-based unsupervised methods is an interesting possibility for future work.

3 Model

Topic modeling is premised on a generative framework in which each word w_t is drawn from a multinomial θ_{y_t}, where y_t is a *hidden topic* indexing the language model that generates w_t. From a modeling standpoint, linear topic segmentation merely adds the constraint that $y_t \in \{y_{t-1}, y_{t-1} + 1\}$. Segmentations that draw boundaries so as to induce compact, low-entropy language models will achieve a

high likelihood. Thus topic models situate lexical cohesion in a probabilistic setting.

For hierarchical segmentation, we take the hypothesis that lexical cohesion is a multi-scale phenomenon. This is represented with a pyramid of language models, shown in Figure 1. Each word may be drawn from any language model above it in the pyramid. Thus, the high-level language models will be required to explain words throughout large parts of the document, while the low-level language models will be required to explain only a local set of words. A hidden variable z_t indicates which level is responsible for generating the word w_t.

Ideally we would like to choose the segmentation $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}} p(\mathbf{w}|\mathbf{y})p(\mathbf{y})$. However, we must deal with the hidden language models Θ and scale-level assignments $\mathbf{z}$. The language models can be integrated out analytically (Section 3.1). Given marginal likelihoods for the hidden variables $\mathbf{z}$, the globally optimal segmentation $\hat{\mathbf{y}}$ can be found using a dynamic program (Section 4.1). Given a segmentation, we can estimate marginals for the hidden variables, using collapsed variational inference (Section 4.2). We iterate between these procedures in an EM-like coordinate-ascent algorithm (Section 4.4) until convergence.

3.1 Language models

We begin the formal presentation of the model with some notation. Each word w_t is modeled as a single draw from a multinomial language model θ_j. The language models in turn are drawn from symmetric Dirichlet distributions with parameter α. The number of language models is written K; the number of words is W; the length of the document is T; and the depth of the hierarchy is L.

For hierarchical segmentation, the vector $\mathbf{y}_t$ indicates the segment index of t at each level of the topic hierarchy; the specific level of the hierarchy responsible for w_t is given by the hidden variable z_t. Thus, $y_t^{(z_t)}$ is the index of the language model that generates w_t.

With these pieces in place, we can write the observation likelihood,

$$p(\mathbf{w}|\mathbf{y}, \mathbf{z}, \Theta) = \prod_t^T p(w_t|\theta_{y_t^{(z_t)}}) = \prod_j^K \prod_{\{t:y_t^{(z_t)}=j\}} p(w_t|\theta_j),$$

where we have merely rearranged the product to group terms that are drawn from the same language model. As the goal is to obtain the hierarchical segmentation and not the language models, the search space can be reduced by marginalizing Θ. The derivation is facilitated by a notational convenience: $\mathbf{x}_j$ represents the lexical counts induced by the set of words $\{w_t : y_t^{(z_t)} = j\}$.

$$\begin{aligned} p(\mathbf{w}|\mathbf{y}, \mathbf{z}, \alpha) &= \prod_j^K \int d\theta_j p(\theta_j|\alpha) p(\mathbf{x}_j|\theta_j) \\ &= \prod_j^K p_{dcm}(\mathbf{x}_j; \alpha) \\ &= \prod_j^K \frac{\Gamma(W\alpha)}{\Gamma(\sum_i^W x_{ji} + \alpha)} \prod_i^W \frac{\Gamma(x_{ji} + \alpha)}{\Gamma(\alpha)}. \end{aligned} \quad (1)$$

Here, p_{dcm} indicates the Dirichlet compound multinomial distribution (Madsen et al., 2005), which is the closed form solution to the integral over language models. Also known as the multivariate Polya distribution, the probability density function can be computed exactly as a ratio of gamma functions. Here we use a symmetric Dirichlet prior α, though asymmetric priors can easily be applied.

Thus far we have treated the hidden variables $\mathbf{z}$ as observed. In fact we will compute approximate marginal probabilities $Q_{z_t}(z_t)$, written $\gamma_{t\ell} \equiv Q_{z_t}(z_t = \ell)$. Writing $\langle x \rangle_{Q_z}$ for the expectation of x under distribution Q_z, we approximate,

$$\langle p_{dcm}(\mathbf{x}_j; \alpha) \rangle_{Q_z} \approx p_{dcm}(\langle \mathbf{x}_j \rangle_{Q_z}; \alpha)$$

$$\langle x_j(i) \rangle_{Q_z} = \sum_{\{t:j \in \mathbf{y}_t\}} \sum_\ell^L \delta(w_t = i)\delta(y_t^{(\ell)} = j)\gamma_{t\ell},$$

where $x_j(i)$ indicates the count for word type i generated from segment j. In the outer sum, we consider all t for possibly drawn from segment j. The inner sum goes over all levels of the pyramid. The delta functions take the value one if the enclosed Boolean expression is true and zero otherwise, so we are adding the fractional counts $\gamma_{t\ell}$ only when $w_t = i$ and $y_t^{(\ell)} = j$.

3.2 Prior on segmentations

Maximizing the joint probability $p(\mathbf{w}, \mathbf{y}) = p(\mathbf{w}|\mathbf{y})p(\mathbf{y})$ leaves the term $p(\mathbf{y})$ as a prior on segmentations. This prior can be used to favor segmentations with the desired granularity. Consider a prior of the form $p(\mathbf{y}) = \prod_{\ell=1}^{L} p(\mathbf{y}^{(\ell)}|\mathbf{y}^{(\ell-1)})$; for notational convenience, we introduce a base level such that $y_t^{(0)} = t$, where every word is a segmentation point. At every level $\ell > 0$, the prior is a Markov process, $p(\mathbf{y}^{(\ell)}|\mathbf{y}^{(\ell-1)}) = \prod_t^T p(y_t^{(\ell)}|y_{t-1}^{(\ell)}, \mathbf{y}^{(\ell-1)})$.

The constraint $y_t^{(\ell)} \in \{y_{t-1}^{(\ell)}, y_{t-1}^{(\ell)} + 1\}$ ensures a linear segmentation at each level. To enforce hierarchical consistency, each $y_t^{(\ell)}$ can be a segmentation point only if t is also a segmentation point at the lower level $\ell - 1$. Zero probability is assigned to segmentations that violate these constraints.

To quantify the prior probability of legal segmentations, assume a set of parameters d_ℓ, indicating the expected segment duration at each level. If t is a valid potential segmentation point at level ℓ (i.e., $y_t^{(\ell-1)} = 1 + y_{t-1}^{(\ell-1)}$), then the prior probability of a segment transition is $r_\ell = d_{\ell-1}/d_\ell$, with $d_0 = 1$. If there are N segments in level ℓ and $M \geq N$ segments in level $\ell - 1$, then the prior $p(\mathbf{y}^{(\ell)}|\mathbf{y}^{(\ell-1)}) = r_\ell^N (1 - r_\ell)^{M-N}$, as long as the hierarchical segmentation constraint is obeyed.

For the purposes of inference it will be preferable to have a prior that decomposes over levels and segments. In particular, we do not want to have to commit to a particular segmentation at level ℓ before segmenting level $\ell + 1$. The above prior can be approximated by replacing M with its expectation $\langle M \rangle_{d_{\ell-1}} = T/d_{\ell-1}$. Then a single segment ranging from w_u to w_v (inclusive) will contribute $\log r_\ell + \frac{v-u}{d_{\ell-1}} \log(1 - r_\ell)$ to the log of the prior.

4 Inference

This section describes the inference for the segmentation $\mathbf{y}$, the approximate marginals Q_Z, and the hyperparameter α.

4.1 Dynamic programming for hierarchical segmentation

While the model structure is reminiscent of a factorial hidden Markov model (HMM), there are important differences that prevent the direct application of HMM inference. Hidden Markov models assume that the parameters of the observation likelihood distributions are available directly, while we marginalize them out. This has the effect of introducing dependencies throughout the state space: the segment assignment for each y_t contributes to lexical counts which in turn affect the observation likelihoods for many other t'. However, due to the left-to-right nature of segmentation, efficient inference of the optimal hierarchical segmentation (given the marginals Q_Z) is still possible.

Let $B^{(\ell)}[u, v]$ represent the log-likelihood of grouping together all contiguous words $w_u \dots w_{v-1}$ at level ℓ of the segmentation hierarchy. Using $\mathbf{x}_t$ to indicate a vector of zeros with one at the position w_t, we can express B more formally:

$$B^{(\ell)}[u, v] = \log p_{dcm}\left(\sum_{t=u}^{v} \mathbf{x}_t \gamma_{t\ell}\right) + \log r_\ell + \frac{v - u - 1}{d_{\ell-1}} \log(1 - r_\ell).$$

The last two terms are from the prior $p(\mathbf{y})$, as explained in Section 3.2. The value of $B^{(\ell)}[u, v]$ is computed for all u, all $v > u$, and all ℓ.

Next, we compute the log-likelihood of the optimal segmentation, which we write as $A^{(L)}[0, T]$. This matrix can be filled in recursively:

$$A^{(\ell)}[u, v] = \max_{u \leq t < v} B^{(\ell)}[t, v] + A^{(\ell-1)}[t, v] + A^{(\ell)}[u, t].$$

The first term adds in the log probability of the segment from t to v at level ℓ. The second term returns the best score for segmenting this same interval at a more detailed level of segmentation. The third term recursively segments the interval from u to t at the same level ℓ. The boundary case $A^{(\ell)}[u, u] = 0$.

4.1.1 Computational complexity

The sizes of A and B are each $\mathcal{O}(LT^2)$. The matrix A can be constructed by iterating through the layers and then iterating: u from 1 to T; v from $u+1$ to T; and t from u to $v+1$. Thus, the time cost for filling A is $\mathcal{O}(LT^3)$. For computing the observation likelihoods in B, the time complexity is $\mathcal{O}(LT^2W)$, where W is the size of the vocabulary – by keeping cumulative lexical counts, we can compute $B[u, v]$ without iterating from u to v.

Eisenstein and Barzilay (2008) describe a dynamic program for linear segmentation with a space complexity of $\mathcal{O}(T)$ and time complexities of $\mathcal{O}(T^2)$ to compute the A matrix and $\mathcal{O}(TW)$ to fill the B matrix.[1] Thus, moving to hierarchical segmentation introduces a factor of TL to the complexity of inference.

4.1.2 Discussion

Intuitively, efficient inference is possible because the location of each segment boundary affects the likelihood of only the adjoining segments at the same level of the hierarchy, and their "children" at the lower levels of the hierarchy. Thus, the observation likelihood at each level decomposes across the segments of the level. This is due to the left-to-right nature of segmentation – in general it is not possible to marginalize the language models and still perform efficient inference in HMMs. The prior (Section 3.2) was designed to decompose across segments – if, for example, $p(\mathbf{y})$ explicitly referenced the total number of segments, inference would be more difficult.

A simpler inference procedure would be a greedy approach that makes a fixed decision about the top-level segmentation, and then applies recursion to achieve segmentation at the lower levels. The greedy approach will not be optimal if the best top-level segmentation leads to unsatisfactory results at the lower levels, or if the lower levels could help to disambiguate high-level segmentation. In contrast, the algorithm presented here maximizes the overall score across all levels of the segmentation hierarchy.

[1] The use of dynamic programming for linear topic segmentation goes back at least to (Heinonen, 1998); however, we are aware of no prior work on dynamic programming for hierarchical segmentation.

4.2 Scale-level marginals

The hidden variable z_t represents the level of the segmentation hierarchy from which the word w_t is drawn. Given language models Θ, each w_t can be thought of as a draw from a Bayesian mixture model, with z_t as the index of the component that generates w_t. However, as we are marginalizing the language models, standard mixture model inference techniques do not apply. One possible solution would be to instantiate the maximum *a posteriori* language models after segmenting, but we would prefer not to have to commit to specific language models. Collapsed Gibbs sampling (Griffiths and Steyvers, 2004) is another possibility, but sampling-based solutions may not be ideal from a performance standpoint.

Recent papers by Teh et al. (2007) and Sung et al. (2008) point to an appealing alternative: collapsed variational inference (called latent-state variational Bayes by Sung et al.). Collapsed variational inference integrates over the parameters (in this case, the language models) and computes marginal distributions for the latent variables, $Q_{\mathbf{z}}$. However, due to the difficulty of computing the expectation of the normalizing term, these marginal probabilities are available only in approximation.

More formally, we wish to compute the approximate distribution $Q_{\mathbf{z}}(\mathbf{z}) = \prod_t^T Q_{z_t}(z_t)$, factorizing across all latent variables. As is typical in variational approaches, we fit this distribution by optimizing a lower bound on the data marginal likelihood $p(\mathbf{w}, \mathbf{z}|\mathbf{y})$ – we condition on the segmentation $\mathbf{y}$ because we are treating it as fixed in this part of the inference. The lower bound can be optimized by iteratively setting,

$$Q_{z_t}(z_t) \propto \exp\left\{\langle \log P(\mathbf{x}, \mathbf{z}|\mathbf{y})\rangle_{\sim Q_{z_t}}\right\},$$

indicating the expectation under $Q_{z_t'}$ for all $t' \neq t$. Due to the couplings across $\mathbf{z}$, it is not possible to compute this expectation directly, so we use the first-order approximation described in (Sung et al., 2008). In this approximation, the value $Q_{z_t}(z_t = \ell)$ – which we abbreviate as $\gamma_{t\ell}$ – takes the form of the likelihood of the observation w_t, given a modified mixture model. The parameters of the mixture model are based on the priors and the counts of $\mathbf{w}$

and γ for all $t' \neq t$:

$$\gamma_{t\ell} \propto \beta_\ell \frac{\tilde{x}_\ell^{\neg t}(w_t)}{\sum_i^W \tilde{x}_\ell^{\neg t}(i)} \quad (2)$$

$$\tilde{x}_\ell^{\neg t}(i) = \alpha_\ell(i) + \sum_{t' \neq t} \gamma_{t'\ell} \delta(w_{t'} = i). \quad (3)$$

The first term in equation 2 is the set of component weights β_ℓ, which are fixed at $1/L$ for all ℓ. The fraction represents the posterior estimate of the language models: standard Dirichlet-multinomial conjugacy gives a sum of counts plus a Dirichlet prior (equation 3). Thus, the form of the update is extremely similar to collapsed Gibbs sampling, except that we maintain the full distribution over z_t rather than sampling a specific value. The derivation of this update is beyond the scope of this paper, but is similar to the mixture of Bernoullis model presented in Section 5 of (Sung et al., 2008).

Iterative updates of this form are applied until the change in the lower bound is less than 10^{-3}. This procedure appears at step 5a of algorithm 1.

4.3 Hyperparameter estimation

The inference procedure defined here includes two parameters: α, the symmetric Dirichlet prior on the language models; and $\mathbf{d}$, the expected segment durations. The granularity of segmentation is considered to be a user-defined characteristic, so there is no "right answer" for how to set this parameter. We simply use the oracle segment durations, and provide the same oracle to the baseline methods where possible. As discussed in Section 6, this parameter had little effect on system performance.

The α parameter controls the expected sparsity of the induced language models; it will be set automatically. Given a segmentation $\mathbf{y}$ and hidden-variable marginals γ, we can maximize $p(\alpha, \mathbf{w}|\mathbf{y}, \gamma) = p_{dcm}(\mathbf{w}|\mathbf{y}, \gamma, \alpha)p(\alpha)$ through gradient descent. The Dirichlet compound multinomial has a tractable gradient, which can be computed using scaled counts, $\tilde{\mathbf{x}}_j = \sum_{t:y_t^{(z_t)}=j} \gamma_{tj}\mathbf{x}_t$ (Minka, 2003). The scaled counts are taken for each segment j across the entire segmentation hierarchy. The likelihood $p(\tilde{\mathbf{x}}|\alpha)$ then has the same form as equation 1, with the x_{ji} terms replaced by $\tilde{x}_{ji}$. The gradient of the log-likelihood is thus a sum across segments,

$$d\ell/d\alpha = \sum_j^K W(\Psi(W\alpha) - \Psi(\alpha)) + \sum_i^W \Psi(\tilde{x}_{ji} + \alpha) - \Psi(W\alpha + \sum_i^W \tilde{x}_{ji}).$$

Here, Ψ indicates the digamma function, which is the derivative of the log gamma function. The prior $p(\alpha)$ takes the form of a Gamma distribution with parameters $\mathcal{G}(1, 1)$, which has the effect of discouraging large values of α. With these parameters, the gradient of the Gamma distribution with respect to α is negative one. To optimize α, we interpose an epoch of L-BFGS (Liu and Nocedal, 1989) optimization after maximizing γ (Step 5b of algorithm 1).

Algorithm 1 Complete segmentation inference

1. **Input** text $\mathbf{w}$; expected durations d.
2. $\gamma \leftarrow$ INITIALIZE-GAMMA($\mathbf{w}$)
3. $\hat{\mathbf{y}} \leftarrow$ EQUAL-WIDTH-SEG($\mathbf{w}, d$)
4. $\alpha \leftarrow .1$
5. **Do**
   (a) $\gamma \leftarrow$ ESTIMATE-GAMMA($\hat{\mathbf{y}}, \mathbf{w}, \gamma, \alpha$)
   (b) $\alpha \leftarrow$ ESTIMATE-ALPHA($\hat{\mathbf{y}}, \mathbf{w}, \gamma$)
   (c) $\mathbf{y} \leftarrow$ SEGMENT($\mathbf{w}, \gamma, \alpha, d$)
   (d) **If** $\mathbf{y} = \hat{\mathbf{y}}$ **then return** $\mathbf{y}$
   (e) **Else** $\hat{\mathbf{y}} \leftarrow \mathbf{y}$

4.4 Combined inference procedure

The final inference procedure alternates between updating the marginals γ, the Dirichlet prior α, and the MAP segmentation $\hat{\mathbf{y}}$. Since the procedure makes hard decisions on α and the segmentations $\mathbf{y}$, it can be thought of as a form of Viterbi expectation-maximization (EM). When a repeated segmentation is encountered, the procedure terminates. Initialization involves constructing a segmentation $\hat{\mathbf{y}}$ in which each level is segmented uniformly, based on the expected segment duration d_ℓ. The hidden variable marginals γ are initialized randomly. While there is no guarantee of finding the global maximum, little sensitivity to the random initialization of γ was observed in preliminary experiments.

The dynamic program described in this section achieves polynomial time complexity, but $\mathcal{O}(LT^3)$

can still be slow when T is the number of word tokens in a large document such as a textbook. For this reason, we only permit segment boundaries to be placed at gold-standard sentence boundaries. The only change to the algorithm is that the tables A and B need contain only cells for each sentence rather than for each word token – hidden variable marginals are still computed for each word token. Implemented in Java, the algorithm runs in roughly five minutes for a document with 1000 sentences on a dual core 2.4 GHz machine.

5 Experimental Setup

Corpora The dataset for evaluation is drawn from a medical textbook (Walker et al., 1990).[2] The text contains 17083 sentences, segmented hierarchically into twelve high-level parts, 150 chapters, and 520 sub-chapter sections. Evaluation is performed separately on each of the twelve parts, with the task of correctly identifying the chapter and section boundaries. Eisenstein and Barzilay (2008) use the same dataset to evaluate linear topic segmentation, though they evaluated only at the level of sections, given gold standard chapter boundaries.

Practical applications of topic segmentation typically relate to more informal documents such as blogs or speech transcripts (Hsueh et al., 2006), as formal texts such as books already contain segmentation markings provided by the author. The premise of this evaluation is that textbook corpora provide a reasonable proxy for performance on less structured data. However, further clarification of this point is an important direction for future research.

Metrics All experiments are evaluated in terms of the commonly-used P_k and WindowDiff metrics (Pevzner and Hearst, 2002). Both metrics pass a window through the document, and assess whether the sentences on the edges of the window are properly segmented with respect to each other. WindowDiff is stricter in that it requires that the number of intervening segments between the two sentences be identical in the hypothesized and the reference segmentations, while P_k only asks whether the two sentences are in the same segment or not. This evaluation uses source code provided by Malioutov and Barzilay (2006).

Experimental system The joint hierarchical Bayesian model described in this paper is called HIERBAYES. It performs a three-level hierarchical segmentation, in which the lowest level is for sub-chapter sections, the middle level is for chapters, and the top level spans the entire part. This top-level has the effect of limiting the influence of words that are common throughout the document.

Baseline systems As noted in Section 2, there is little related work on unsupervised hierarchical segmentation. However, a straightforward baseline is a greedy approach: first segment at the top level, and then recursively feed each top-level segment to the segmenter again. Any linear segmenter can be plugged into this baseline as a "black box."

To isolate the contribution of joint inference, the greedy framework can be combined with a one-level version of the Bayesian segmentation algorithm described here. This is equivalent to BAYESSEG, which achieved the best reported performance on the linear segmentation of this same dataset (Eisenstein and Barzilay, 2008). The hierarchical segmenter built by placing BAYESSEG in a greedy algorithm is called GREEDY-BAYES.

To identify the contribution of the Bayesian segmentation framework, we can plug in alternative linear segmenters. Two frequently-cited systems are LCSEG (Galley et al., 2003) and TEXTSEG (Utiyama and Isahara, 2001). LCSEG optimizes a metric of lexical cohesion based on lexical chains. TEXTSEG employs a probabilistic segmentation objective that is similar to ours, but uses maximum *a posteriori* estimates of the language models, rather than marginalizing them out. Other key differences are that they set $\alpha = 1$, and use a minimum description length criterion to determine segmentation granularity. Both of these baselines were run using their default parametrization.

Finally, as a minimal baseline, UNIFORM produces a hierarchical segmentation with the ground truth number of segments per level and uniform duration per segment at each level.

Preprocessing The Porter (1980) stemming algorithm is applied to group equivalent lexical items. A set of stop-words is also removed, using the same

[2] The full text of this book is available for free download at http://onlinebooks.library.upenn.edu.

	chapter			section			average	
	# segs	P_k	WD	# segs	P_k	WD	P_k	WD
HIERBAYES	5.0	**.248**	**.255**	8.5	.312	.351	.280	**.303**
GREEDY-BAYES	19.0	.260	.372	19.5	**.275**	**.340**	**.268**	.356
GREEDY-LCSEG	7.8	.256	.286	52.2	.351	.455	.304	.371
GREEDY-TEXTSEG	11.5	.251	.277	88.4	.473	.630	.362	.454
UNIFORM	**12.5**	.487	.491	**43.3**	.505	.551	.496	.521

Table 1: Segmentation accuracy and granularity. Both the P_k and WindowDiff (WD) metrics are penalties, so lower scores are better. The **# segs** columns indicate the average number of segments at each level; the gold standard segmentation granularity is given in the UNIFORM row, which obtains this granularity by construction.

list originally employed by several competitive systems (Utiyama and Isahara, 2001).

6 Results

Table 1 presents performance results for the joint hierarchical segmenter and the three greedy baselines. As shown in the table, the hierarchical system achieves the top overall performance on the harsher WindowDiff metric. In general, the greedy segmenters each perform well at one of the two levels and poorly at the other level. The joint hierarchical inference of HIERBAYES enables it to achieve balanced performance at the two levels.

The GREEDY-BAYES system achieves a slightly better average P_k than HIERBAYES, but has a very large gap between its P_k and WindowDiff scores. The P_k metric requires only that the system correctly classify whether two points are in the same or different segments, while the WindowDiff metric insists that the exact number of interposing segments be identified correctly. Thus, the generation of spurious short segments may explain the gap between the metrics.

LCSEG and TEXTSEG use heuristics to determine segmentation granularity; even though these methods did not score well in terms of segmentation accuracy, they were generally closer to the correct granularity. In the Bayesian methods, granularity is enforced by the Markov prior described in Section 3.2. This prior was particularly ineffective for GREEDY-BAYES, which gave nearly the same number of segments at both levels, despite the different settings of the expected duration parameter d.

The Dirichlet prior α was selected automatically, but informal experiments with manual settings suggest that this parameter exerts a stronger influence on segmentation granularity. Low settings reflect an expectation of sparse lexical counts and thus encourage short segments, while high settings reflect an expectation of evenly-distributed counts and thus lead to long segments. Further investigation is needed on how best to control segmentation granularity in a Bayesian setting.

7 Discussion

While it is widely agreed that language often displays hierarchical topic structure (Grosz, 1977), there have been relatively few attempts to extract such structure automatically. This paper shows that the lexical features that have been successfully exploited in linear segmentation can also be used to extract a hierarchical segmentation, due to the phenomenon of multi-scale lexical cohesion. The Bayesian methodology offers a principled probabilistic formalization of multi-scale cohesion, yielding an accurate and fast segmentation algorithm with a minimal set of tunable parameters.

It is interesting to consider how multi-scale segmentation might be extended to finer-grain segments, such as paragraphs. The lexical counts at the paragraph level will be sparse, so lexical cohesion alone is unlikely to be sufficient. Rather it may be necessary to model discourse connectors and lexical semantics explicitly. The development of more comprehensive Bayesian models for discourse structure seems an exciting direction for future research.

Acknowledgments Thanks to Michel Galley, Igor Malioutov, and Masao Utiyama for making their topic segmentation systems publicly available, and to the anonymous reviewers for useful feedback. This research is supported by the Beckman Postdoctoral Fellowship.

References

David M. Blei and Pedro J. Moreno. 2001. Topic segmentation with an aspect hidden markov model. In *SIGIR*, pages 343–348.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of EMNLP*.

Micha Elsner and Eugene Charniak. 2008. You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement. In *Proceedings of ACL*.

Michel Galley, Katheen McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. pages 562–569.

T.L. Griffiths and M. Steyvers. 2004. Finding scientific topics.

Barbara Grosz and Candace Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Barbara Grosz. 1977. The representation and use of focus in dialogue understanding. Technical Report 151, Artificial Intelligence Center, SRI International.

M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman.

Marti A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of ACL*, pages 9–16.

Oskari Heinonen. 1998. Optimal Multi-Paragraph Text Segmentation by Dynamic Programming. In *Proceedings of ACL*, pages 1484–1486.

P.Y. Hsueh, J. Moore, and S. Renals. 2006. Automatic segmentation of multiparty dialogue. In *Proccedings of EACL*.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528.

R.E. Madsen, D. Kauchak, and C. Elkan. 2005. Modeling word burstiness using the Dirichlet distribution. In *Proceedings of ICML*.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of ACL*, pages 25–32.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8:243–281.

Thomas P. Minka. 2003. Estimating a dirichlet distribution. Technical report, Massachusetts Institute of Technology.

Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14:130–137.

M. Purver, T.L. Griffiths, K.P. Körding, and J.B. Tenenbaum. 2006. Unsupervised topic modelling for multiparty spoken discourse. In *Proceedings of ACL*, pages 17–24.

Jaemo Sung, Zoubin Ghahramani, and Sung-Yang Bang. 2008. Latent-space variational bayes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(12):2236–2242, Dec.

Y.W. Teh, D. Newman, and M. Welling. 2007. A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation. In *NIPS*, volume 19, page 1353.

Masao Utiyama and Hitoshi Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of ACL*, pages 491–498.

H. Kenneth Walker, W. Dallas Hall, and J. Willis Hurst, editors. 1990. *Clinical Methods : The History, Physical, and Laboratory Examinations*. Butterworths.

Y. Yaari. 1997. Segmentation of Expository Texts by Hierarchical Agglomerative Clustering. In *Recent Advances in Natural Language Processing*.

Exploring Content Models for Multi-Document Summarization

Aria Haghighi
UC Berkeley, CS Division
aria42@cs.berkeley.edu

Lucy Vanderwende
Microsoft Research
Lucy.Vanderwende@microsoft.com

Abstract

We present an exploration of generative probabilistic models for multi-document summarization. Beginning with a simple word frequency based model (Nenkova and Vanderwende, 2005), we construct a sequence of models each injecting more structure into the representation of document set content and exhibiting ROUGE gains along the way. Our final model, HIERSUM, utilizes a hierarchical LDA-style model (Blei et al., 2004) to represent content specificity as a hierarchy of topic vocabulary distributions. At the task of producing generic DUC-style summaries, HIERSUM yields state-of-the-art ROUGE performance and in pairwise user evaluation strongly outperforms Toutanova et al. (2007)'s state-of-the-art discriminative system. We also explore HIERSUM's capacity to produce multiple 'topical summaries' in order to facilitate content discovery and navigation.

1 Introduction

Over the past several years, there has been much interest in the task of multi-document summarization. In the common Document Understanding Conference (DUC) formulation of the task, a system takes as input a document set as well as a short description of desired summary focus and outputs a word length limited summary.[1] To avoid the problem of generating cogent sentences, many systems opt for an extractive approach, selecting sentences from the document set which best reflect its core content.[2]

There are several approaches to modeling document content: simple word frequency-based methods (Luhn, 1958; Nenkova and Vanderwende, 2005), graph-based approaches (Radev, 2004; Wan and Yang, 2006), as well as more linguistically motivated techniques (Mckeown et al., 1999; Leskovec et al., 2005; Harabagiu et al., 2007). Another strand of work (Barzilay and Lee, 2004; Daumé III and Marcu, 2006; Eisenstein and Barzilay, 2008), has explored the use of structured probabilistic topic models to represent document content. However, little has been done to directly compare the benefit of complex content models to simpler surface ones for generic multi-document summarization.

In this work we examine a series of content models for multi-document summarization and argue that LDA-style probabilistic topic models (Blei et al., 2003) can offer state-of-the-art summarization quality as measured by automatic metrics (see section 5.1) and manual user evaluation (see section 5.2). We also contend that they provide convenient building blocks for adding more structure to a summarization model. In particular, we utilize a variation of the hierarchical LDA topic model (Blei et al., 2004) to discover multiple specific 'subtopics' within a document set. The resulting model, HIERSUM (see section 3.4), can produce general summaries as well as summaries for any of the learned sub-topics.

2 Experimental Setup

The task we will consider is extractive multi-document summarization. In this task we assume a document collection $\mathcal{D}$ consisting of documents $D_1, \ldots, D_n$ describing the same (or closely related)

[1] In this work, we ignore the summary focus. Here, the word *topic* will refer to elements of our statistical model rather than summary focus.

[2] Note that sentence extraction does not solve the problem of selecting and ordering summary sentences to form a coherent narrative (Lapata, 2003).

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 362–370,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

set of events. Our task will be to propose a summary $\mathbf{S}$ consisting of sentences in $\mathcal{D}$ totaling at most L words.[3] Here as in much extractive summarization, we will view each sentence as a bag-of-words or more generally a bag-of-ngrams (see section 5.1). The most prevalent example of this data setting is document clusters found on news aggregator sites.

2.1 Automated Evaluation

For model development we will utilize the DUC 2006 evaluation set[4] consisting of 50 document sets each with 25 documents; final evaluation will utilize the DUC 2007 evaluation set (section 5).

Automated evaluation will utilize the standard DUC evaluation metric ROUGE (Lin, 2004) which represents recall over various n-grams statistics from a system-generated summary against a set of human-generated peer summaries.[5] We compute ROUGE scores with and without stop words removed from peer and proposed summaries. In particular, we utilize R-1 (recall against unigrams), R-2 (recall against bigrams), and R-SU4 (recall against skip-4 bigrams)[6]. We present R-2 without stop words in the running text, but full development results are presented in table 1. Official DUC scoring utilizes the jackknife procedure and assesses significance using bootstrapping resampling (Lin, 2004). In addition to presenting automated results, we also present a user evaluation in section 5.2.

3 Summarization Models

We present a progression of models for multi-document summarization. Inference details are given in section 4.

3.1 SumBasic

The SUMBASIC algorithm, introduced in Nenkova and Vanderwende (2005), is a simple effective procedure for multi-document extractive summarization. Its design is motivated by the observation that the relative frequency of a non-stop word in a document set is a good predictor of a word appearing in a human summary. In SUMBASIC, each sentence S is assigned a score reflecting how many high-frequency words it contains,

$$Score(S) = \sum_{w \in S} \frac{1}{|S|} P_{\mathcal{D}}(w) \quad (1)$$

where $P_{\mathcal{D}}(\cdot)$ initially reflects the observed unigram probabilities obtained from the document collection $\mathcal{D}$. A summary $\mathbf{S}$ is progressively built by adding the highest scoring sentence according to (1).[7]

In order to discourage redundancy, the words in the selected sentence are updated $P_{\mathcal{D}}^{new}(w) \propto P_{\mathcal{D}}^{old}(w)^2$. Sentences are selected in this manner until the summary word limit has been reached.

Despite its simplicity, SUMBASIC yields 5.3 R-2 without stop words on DUC 2006 (see table 1).[8] By comparison, the highest-performing ROUGE system at the DUC 2006 evaluation, SUMFOCUS, was built on top of SUMBASIC and yielded a 6.0, which is not a statistically significant improvement (Vanderwende et al., 2007).[9]

Intuitively, SUMBASIC is trying to select a summary which has sentences where most words have high likelihood under the document set unigram distribution. One conceptual problem with this objective is that it inherently favors repetition of frequent non-stop words despite the 'squaring' update. Ideally, a summarization criterion should be more recall oriented, penalizing summaries which omit moderately frequent document set words and quickly diminishing the reward for repeated use of word.

Another more subtle shortcoming is the use of the raw empirical unigram distribution to represent content significance. For instance, there is no distinction between a word which occurs many times in the same document or the same number of times across several documents. Intuitively, the latter word is more indicative of significant document set content.

3.2 KLSum

The KLSUM algorithm introduces a criterion for selecting a summary $\mathbf{S}$ given document collection $\mathcal{D}$,

$$\mathbf{S}^* = \min_{\mathbf{S}: words(\mathbf{S}) \leq L} KL(P_{\mathcal{D}} \| P_{\mathbf{S}}) \quad (2)$$

[3] For DUC summarization tasks, L is typically 250.

[4] http://www-nlpir.nist.gov/projects/duc/data.html

[5] All words from peer and proposed summaries are lowercased and stemmed.

[6] Bigrams formed by skipping at most two words.

[7] Note that sentence order is determined by the order in which sentences are selected according to (1).

[8] This result is presented as 0.053 with the official ROUGE scorer (Lin, 2004). Results here are scaled by 1,000.

[9] To be fair obtaining statistical significance in ROUGE scores is quite difficult.

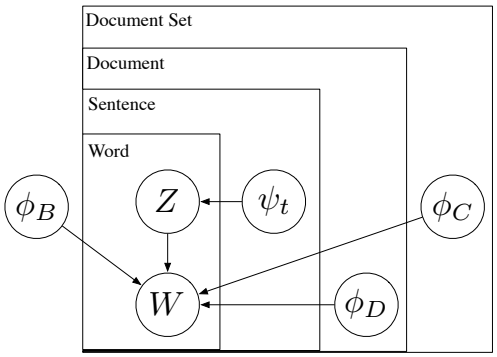


Figure 1: Graphical model depiction of TOPICSUM model (see section 3.3). Note that many hyperparameter dependencies are omitted for compactness.

where $P_{\mathbf{S}}$ is the empirical unigram distribution of the candidate summary $\mathbf{S}$ and $KL(P\|Q)$ represents the Kullback-Lieber (KL) divergence given by $\sum_w P(w) \log \frac{P(w)}{Q(w)}$.[10] This quantity represents the divergence between the true distribution P (here the document set unigram distribution) and the approximating distribution Q (the summary distribution). This criterion casts summarization as finding a set of summary sentences which closely match the document set unigram distribution. Lin et al. (2006) propose a related criterion for robust summarization evaluation, but to our knowledge this criteria has been unexplored in summarization systems. We address optimizing equation (2) as well as summary sentence ordering in section 4.

KLSUM yields 6.0 R-2 without stop words, beating SUMBASIC but not with statistical significance. It is worth noting however that KLSUM's performance matches SUMFOCUS (Vanderwende et al., 2007), the highest R-2 performing system at DUC 2006.

3.3 TopicSum

As mentioned in section 3.2, the raw unigram distribution $P_{\mathcal{D}}(\cdot)$ may not best reflect the content of $\mathcal{D}$ for the purpose of summary extraction. We propose TOPICSUM, which uses a simple LDA-like topic model (Blei et al., 2003) similar to Daumé III and Marcu (2006) to estimate a content distribution for summary extraction.[11] We extract summary sentences as before using the KLSUM criterion (see equation (2)), plugging in a learned content distribution in place of the raw unigram distribution.

System	ROUGE -stop			ROUGE all		
	R-1	R-2	R-SU4	R-1	R-2	R-SU4
SUMBASIC	29.6	5.3	8.6	36.1	7.1	12.3
KLSUM	30.6	6.0	8.9	38.9	8.3	13.7
TOPICSUM	**31.7**	**6.3**	9.1	39.2	8.4	13.6
HIERSUM	30.5	**6.4**	9.2	40.1	8.6	**14.3**

Table 1: ROUGE results on DUC2006 for models presented in section 3. Results in bold represent results statistically significantly different from SUMBASIC in the appropriate metric.

First, we describe our topic model (see figure 1) which generates a collection of document sets. We assume a fixed vocabulary V:[12]

1. Draw a background vocabulary distribution ϕ_B from DIRICHLET(V,λ_B) shared across document collections[13] representing the background distribution over vocabulary words. This distribution is meant to flexibly model stop words which do not contribute content. We will refer to this topic as BACKGROUND.

2. For each document set $\mathcal{D}$, we draw a content distribution ϕ_C from DIRICHLET(V,λ_C) representing the significant content of $\mathcal{D}$ that we wish to summarize. We will refer to this topic as CONTENT.

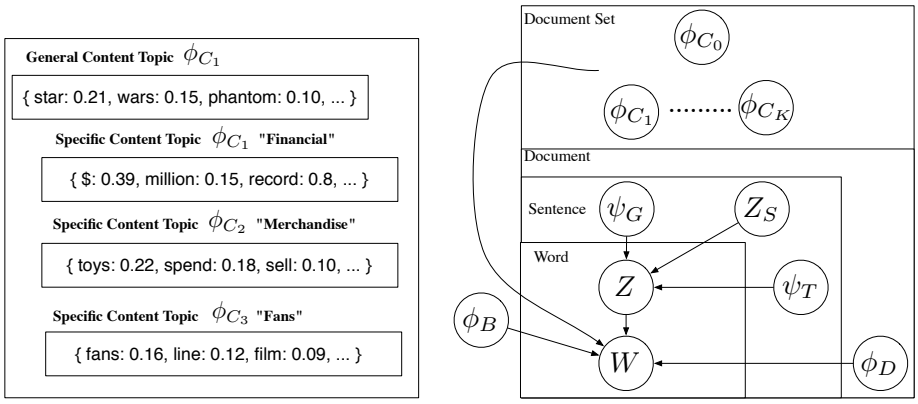
3. For each document D in $\mathcal{D}$, we draw a document-specific vocabulary distribution ϕ_D from DIRICHLET(V,λ_D) representing words which are local to a single document, but do not appear across several documents. We will refer to this topic as DOCSPECIFIC.

[10] In order to ensure finite values of KL-divergence we smoothe $P_{\mathbf{S}}(\cdot)$ so that it has a small amount of mass on all document set words.

[11] A topic model is a probabilistic generative process that generates a collection of documents using a mixture of topic vocabulary distributions (Steyvers and Griffiths, 2007). Note this usage of topic is unrelated to the summary focus given for document collections; this information is ignored by our models.

[12] In contrast to previous models, stop words are *not* removed in pre-processing.

[13] DIRICHLET(V,λ) represents the symmetric Dirichlet prior distribution over V each with a pseudo-count of λ. Concrete pseudo-count values will be given in section 4.



(a) Content Distributions (b) HIERSUM Graphical Model

Figure 2: (a): Examples of general versus specific content distributions utilized by HIERSUM (see section 3.4). The general content distribution ϕ_{C_0} will be used throughout a document collection and represents core concepts in a story. The specific content distributions represent topical 'sub-stories' with vocabulary tightly clustered together but consistently used across documents. Quoted names of specific topics are given manually to facilitate interpretation. (b) Graphical model depiction of the HIERSUM model (see section 3.4). Similar to the TOPICSUM model (see section 3.3) except for adding complexity in the content hierarchy as well as sentence-specific prior distributions between general and specific content topics (early sentences should have more general content words). Several dependencies are missing from this depiction; crucially, each sentence's specific topic Z_S depends on the last sentence's Z_S.

4. For each sentence S of each document D, draw a distribution ψ_T over topics (CONTENT, DOCSPECIFIC, BACKGROUND) from a Dirichlet prior with pseudo-counts $(1.0, 5.0, 10.0)$.[14] For each word position in the sentence, we draw a topic Z from ψ_T, and a word W from the topic distribution Z indicates.

Our intent is that ϕ_C represents the core content of a document set. Intuitively, ϕ_C does not include words which are common amongst several document collections (modeled with the BACKGROUND topic), or words which don't appear across many documents (modeled with the DOCSPECIFIC topic). Also, because topics are tied together at the sentence level, words which frequently occur with other content words are more likely to be considered content words.

We ran our topic model over the DUC 2006 document collections and estimated the distribution $\phi_C(\cdot)$ for each document set.[15] Then we extracted a summary using the KLSUM criterion with our estimated ϕ_C in place of the the raw unigram distribution. Doing so yielded 6.3 R-2 without stop words (see TOPICSUM in table 1); while not a statistically significant improvement over KLSUM, it is our first model which outperforms SUMBASIC with statistical significance.

Daumé III and Marcu (2006) explore a topic model similar to ours for query-focused multi-document summarization.[16] Crucially however, Daumé III and Marcu (2006) selected sentences with the highest expected number of CONTENT words.[17] We found that in our model using this extraction criterion yielded 5.3 R-2 without stop words, significantly underperforming our TOPICSUM model. One reason for this may be that Daumé III and Marcu (2006)'s criterion encourages selecting sentences which have words that are confidently generated by the CONTENT distribution, but not necessarily sentences which contain a plurality of it's mass.

[14] The different pseudo-counts reflect the intuition that most of the words in a document come from the BACKGROUND and DOCSPECIFIC topics.

[15] While possible to obtain the predictive posterior CONTENT distribution by analytically integrating over ϕ_C (Blei et al., 2003), doing so gave no benefit.

[16] Daumé III and Marcu (2006) note their model could be used outside of query-focused summarization.

[17] This is phrased as selecting the sentence which has the highest posterior probability of emitting CONTENT topic words, but this is equivalent.

(a) HIERSUM output

The French government Saturday announced several emergency measures to support the jobless people, including sending an additional 500 million franc (84 million U.S. dollars) unemployment aid package. The unemployment rate in France dropped by 0.3 percent to stand at 12.4 percent in November, said the Ministry of Employment Tuesday.

(b) PYTHY output

Several hundred people took part in the demonstration here today against the policies of the world's most developed nations. The 12.5 percent unemployment rate is haunting the Christmas season in France as militants and unionists staged several protests over the past week against unemployment.

(c) Ref output

High unemployment is France's main economic problem, despite recent improvements. A top worry of French people, it is a factor affecting France's high suicide rate. Long-term unemployment causes social exclusion and threatens France's social cohesion.

(d) Reference Unigram Coverage

word	Ref	PYTHY	HIERSUM
unemployment	8	9	10
france's	6	1	4
francs	4	0	1
high	4	1	2
economic	2	0	1
french	2	1	3
problem	2	0	1
benefits	2	0	0
social	2	0	2
jobless	2	1	2

Table 2: Example summarization output for systems compared in section 5.2. (a), (b), and (c) represent the first two sentences output from PYTHY, HIERSUM, and reference summary respectively. In (d), we present the most frequent non-stop unigrams appearing in the reference summary and their counts in the PYTHY and HIERSUM summaries. Note that many content words in the reference summary absent from PYTHY's proposal are present in HIERSUM's.

3.4 HIERSUM

Previous sections have treated the content of a document set as a single (perhaps learned) unigram distribution. However, as Barzilay and Lee (2004) observe, the content of document collections is highly structured, consisting of several topical themes, each with its own vocabulary and ordering preferences. For concreteness consider the DUC 2006 document collection describing the opening of *Star Wars: Episode 1* (see figure 2(a)).

While there are words which indicate the general content of this document collection (e.g. *star*, *wars*), there are several sub-stories with their own specific vocabulary. For instance, several documents in this collection spend a paragraph or two talking about the financial aspect of the film's opening and use a specific vocabulary there (e.g. *$*, *million*, *record*). A user may be interested in general content of a document collection or, depending on his or her interests, one or more of the sub-stories. We choose to adapt our topic modeling approach to allow modeling this aspect of document set content.

Rather than drawing a single CONTENT distribution ϕ_C for a document collection, we now draw a general content distribution ϕ_{C_0} from DIRICHLET(V,λ_G) as well as specific content distributions ϕ_{C_i} for $i = 1, \ldots, K$ each from DIRICHLET(V,λ_S).[18] Our intent is that ϕ_{C_0} represents the general content of the document collection and each ϕ_{C_i} represents specific sub-stories.

As with TOPICSUM, each sentence has a distribution ψ_T over topics (BACKGROUND, DOCSPECIFIC, CONTENT). When BACKGROUND or DOCSPECIFIC topics are chosen, the model works exactly as in TOPICSUM. However when the CONTENT topic is drawn, we must decide whether to emit a general content word (from ϕ_{C_0}) or from one of the specific content distributions (from one of ϕ_{C_i} for $i = 1, \ldots, K$). The generative story of TOPICSUM is altered as follows in this case:

- **General or Specific?** We must first decide whether to use a general or specific content word. Each sentence draws a binomial distribution ψ_G determining whether a CONTENT word in the sentence will be drawn from the general or a specific topic distribution. Reflecting the intuition that the earlier sentences in a document[19] describe the general content of a story, we bias ψ_G to be drawn from BETA(5,2), preferring general content words, and every later sentence from BETA(1,2).[20]

- **What Specific Topic?** If ψ_G decides we are

[18] We choose K=3 in our experiments, but one could flexibly choose K as Blei et al. (2004) does.

[19] In our experiments, the first 5 sentences.

[20] BETA(a,b) represents the beta prior over binomial random variables with a and b being pseudo-counts for the first and second outcomes respectively.

emitting a topic specific content word, we must decide which of $\phi_{C_1}, \ldots, \phi_{C_K}$ to use. In order to ensure tight lexical cohesion amongst the specific topics, we assume that each sentence draws a single specific topic Z_S used for every specific content word in that sentence. Reflecting intuition that adjacent sentences are likely to share specific content vocabulary, we utilize a 'sticky' HMM as in Barzilay and Lee (2004) over the each sentences' Z_S. Concretely, Z_S for the first sentence in a document is drawn uniformly from $1, \ldots, K$, and each subsequent sentence's Z_S will be identical to the previous sentence with probability σ, and with probability $1 - \sigma$ we select a successor topic from a learned transition distribution amongst $1, \ldots, K$.[21]

Our intent is that the general content distribution ϕ_{C_0} now prefers words which not only appear in many documents, but also words which appear consistently throughout a document rather than being concentrated in a small number of sentences. Each specific content distribution ϕ_{C_i} is meant to model topics which are used in several documents but tend to be used in concentrated locations.

HIERSUM can be used to extract several kinds of summaries. It can extract a general summary by plugging ϕ_{C_0} into the KLSUM criterion. It can also produce topical summaries for the learned specific topics by extracting a summary over each ϕ_{C_i} distribution; this might be appropriate for a user who wants to know more about a particular substory. While we found the general content distribution (from ϕ_{C_0}) to produce the best single summary, we experimented with utilizing topical summaries for other summarization tasks (see section 6.1). The resulting system, HIERSUM yielded 6.4 R-2 without stop words. While not a statistically significant improvement in ROUGE over TOPICSUM, we found the summaries to be noticeably improved.

4 Inference and Model Details

Since globally optimizing the KLSUM criterion in equation (equation (2)) is exponential in the total number of sentences in a document collection, we opted instead for a simple approximation where sentences are greedily added to a summary so long as they decrease KL-divergence. We attempted more complex inference procedures such as McDonald (2007), but these attempts only yielded negligible performance gains. All summary sentence ordering was determined as follows: each sentence in the proposed summary was assigned a number in $[0, 1]$ reflecting its relative sentence position in its source document, and sorted by this quantity.

All topic models utilize Gibbs sampling for inference (Griffiths, 2002; Blei et al., 2004). In general for concentration parameters, the more specific a distribution is meant to be, the smaller its concentration parameter. Accordingly for TOPICSUM, $\lambda_G = \lambda_D = 1$ and $\lambda_C = 0.1$. For HIERSUM we used $\lambda_G = 0.1$ and $\lambda_S = 0.01$. These parameters were minimally tuned (without reference to ROUGE results) in order to ensure that all topic distribution behaved as intended.

5 Formal Experiments

We present formal experiments on the DUC 2007 data main summarization task, proposing a general summary of at most 250 words[22] which will be evaluated automatically and manually in order to simulate as much as possible the DUC evaluation environment.[23] DUC 2007 consists of 45 document sets, each consisting of 25 documents and 4 human reference summaries.

We primarily evaluate the HIERSUM model, extracting a single summary from the general content distribution using the KLSUM criterion (see section 3.2). Although the differences in ROUGE between HIERSUM and TOPICSUM were minimal, we found HIERSUM summary quality to be stronger.

In order to provide a reference for ROUGE and manual evaluation results, we compare against PYTHY, a state-of-the-art supervised sentence extraction summarization system. PYTHY uses human-generated summaries in order to train a sentence ranking system which discriminatively maximizes

[21] We choose $\sigma = 0.75$ in our experiments.

[22] Since the ROUGE evaluation metric is recall-oriented, it is always advantageous - with respect to ROUGE - to use all 250 words.

[23] Although the DUC 2007 main summarization task provides an indication of user intent through topic focus queries, we ignore this aspect of the data.

System	ROUGE w/o stop			ROUGE w/ stop		
	R-1	R-2	R-SU4	R-1	R-2	R-SU4
HIERSUM unigram	34.6	7.3	10.4	43.1	9.7	15.3
HIERSUM bigram	33.8	9.3	11.6	42.4	11.8	16.7
PYTHY w/o simp	34.7	8.7	11.8	42.7	11.4	16.5
PYTHY w/ simp	35.7	8.9	12.1	42.6	11.9	16.8

Table 3: Formal ROUGE experiment results on DUC 2007 document set collection (see section 5.1). While HIERSUM unigram underperforms both PYTHY systems in statistical significance (for R-2 and RU-4 with and without stop words), HIERSUM bigram's performance is comparable and statistically no worse.

ROUGE scores. PYTHY uses several features to rank sentences including several variations of the SUMBASIC score (see section 3.1). At DUC 2007, PYTHY was ranked first overall in automatic ROUGE evaluation and fifth in manual content judgments. As PYTHY utilizes a sentence simplification component, which we do not, we also compare against PYTHY without sentence simplification.

5.1 ROUGE Evaluation

ROUGE results comparing variants of HIERSUM and PYTHY are given in table 3. The HIERSUM system as described in section 3.4 yields 7.3 R-2 without stop words, falling significantly short of the 8.7 that PYTHY without simplification yields. Note that R-2 is a measure of bigram recall and HIERSUM does not represent bigrams whereas PYTHY includes several bigram and higher order n-gram statistics.

In order to put HIERSUM and PYTHY on equal-footing with respect to R-2, we instead ran HIERSUM with each sentence consisting of a bag of bigrams instead of unigrams.[24] All the details of the model remain the same. Once a general content distribution over bigrams has been determined by hierarchical topic modeling, the KLSUM criterion is used as before to extract a summary. This system, labeled HIERSUM bigram in table 3, yields 9.3 R-2 without stop words, significantly outperforming HIERSUM unigram. This model outperforms PYTHY with and without sentence simplification, but not with statistical significance. We conclude that both PYTHY variants and HIERSUM bigram are comparable with respect to ROUGE performance.

[24]Note that by doing topic modeling in this way over bigrams, our model becomes degenerate as it can generate inconsistent bags of bigrams. Future work may look at topic models over n-grams as suggested by Wang et al. (2007).

Question	PYTHY	HIERSUM
Overall	20	49
Non-Redundancy	21	48
Coherence	15	54
Focus	28	41

Table 4: Results of manual user evaluation (see section 5.2). 15 participants expressed 69 pairwise preferences between HIERSUM and PYTHY. For all attributes, HIERSUM outperforms PYTHY; all results are statistically significant as determined by pairwise t-test.

5.2 Manual Evaluation

In order to obtain a more accurate measure of summary quality, we performed a simple user study. For each document set in the DUC 2007 collection, a user was given a reference summary, a PYTHY summary, and a HIERSUM summary;[25] note that the original documents in the set were *not* provided to the user, only a reference summary. For this experiment we use the bigram variant of HIERSUM and compare it to PYTHY without simplification so both systems have the same set of possible output summaries.

The reference summary for each document set was selected according to highest R-2 without stop words against the remaining peer summaries. Users were presented with 4 questions drawn from the DUC manual evaluation guidelines:[26] (1) Overall quality: Which summary was better overall? (2) Non-Redundancy: Which summary was less redundant? (3) Coherence: Which summary was more coherent? (4) Focus: Which summary was more

[25]The system identifier was of course not visible to the user. The order of automatic summaries was determined randomly.

[26]http://www-nlpir.nist.gov/projects/duc/duc2007/quality-questions.txt

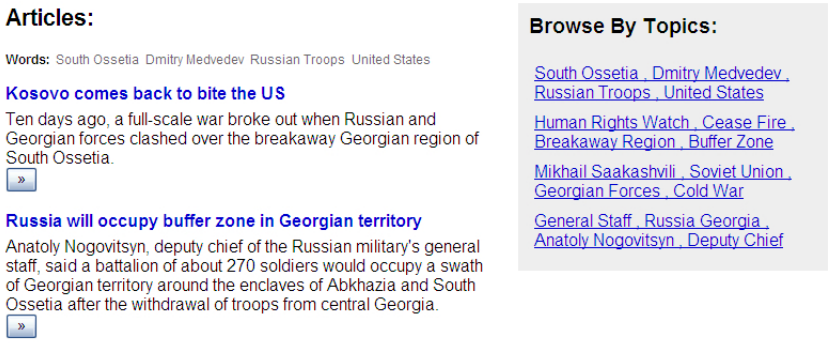


Figure 3: Using HIERSUM to organize content of document set into topics (see section 6.1). The sidebar gives key phrases salient in each of the specific content topics in HIERSUM (see section 3.4). When a topic is clicked in the right sidebar, the main frame displays an extractive 'topical summary' with links into document set articles. Ideally, a user could use this interface to quickly find content in a document collection that matches their interest.

focused in its content, not conveying irrelevant details? The study had 16 users and each was asked to compare five summary pairs, although some did fewer. A total of 69 preferences were solicited. Document collections presented to users were randomly selected from those evaluated fewest.

As seen in table 5.2, HIERSUM outperforms PYTHY under all questions. All results are statistically significant as judged by a simple pairwise t-test with 95% confidence. It is safe to conclude that users in this study strongly preferred the HIERSUM summaries over the PYTHY summaries.

6 Discussion

While it is difficult to qualitatively compare one summarization system over another, we can broadly characterize HIERSUM summaries compared to some of the other systems discussed. For example output from HIERSUM and PYTHY see table 2. On the whole, HIERSUM summaries appear to be significantly less redundant than PYTHY and moderately less redundant than SUMBASIC. The reason for this might be that PYTHY is discriminatively trained to maximize ROUGE which does not directly penalize redundancy. Another tendency is for HIERSUM to select longer sentences typically chosen from an early sentence in a document. As discussed in section 3.4, HIERSUM is biased to consider early sentences in documents have a higher proportion of general content words and so this tendency is to be expected.

6.1 Content Navigation

A common concern in multi-document summarization is that without any indication of user interest or intent providing a single satisfactory summary to a user may not be feasible. While many variants of the general summarization task have been proposed which utilize such information (Vanderwende et al., 2007; Nastase, 2008), this presupposes that a user knows enough of the content of a document collection in order to propose a query.

As Leuski et al. (2003) and Branavan et al. (2007) suggest, a document summarization system should facilitate content discovery and yield summaries relevant to a user's interests. We may use HIERSUM in order to facilitate content discovery via presenting a user with salient words or phrases from the specific content topics parametrized by $\phi_{C_1}, \ldots, \phi_{C_K}$ (for an example see figure 3). While these topics are not adaptive to user interest, they typically reflect lexically coherent vocabularies.

Conclusion

In this paper we have presented an exploration of content models for multi-document summarization and demonstrated that the use of structured topic models can benefit summarization quality as measured by automatic and manual metrics.

Acknowledgements The authors would like to thank Bob Moore, Chris Brockett, Chris Quirk, and Kristina Toutanova for their useful discussions as well as the reviewers for their helpful comments.

References

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *JMLR*.

David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. In *NIPS*.

S.R.K. Branavan, Pawan Deshpande, and Regina Barzilay. 2007. Generating a table-of-contents. In *ACL*.

Hal Daumé III and Daniel Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *EMNLP-SIGDAT*.

Thomas Griffiths. 2002. Gibbs sampling in the generative model of latent dirichlet allocation.

Sanda Harabagiu, Andrew Hickl, and Finley Lacatusu. 2007. Satisfying information needs with multi-document summaries. *Inf. Process. Manage.*, 43(6).

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *ACL*.

Jurij Leskovec, Natasa Milic-frayling, and Marko Grobelnik. 2005. Impact of linguistic analysis on the semantic graph coverage and learning of document extracts. In *In AAAI 05*.

Anton Leuski, Chin-Yew Lin, and Eduard Hovy. 2003. ineats: Interactive multi-document summarization. In *ACL*.

Chin-Yew Lin, Guihong Cao, Jianfeng Gao, and Jian-Yun Nie. 2006. An information-theoretic approach to automatic evaluation of summaries. In *HLT-NAACL*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*.

H.P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal*.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *ECIR*.

Kathleen R. Mckeown, Judith L. Klavans, Vasileios Hatzivassiloglou, Regina Barzilay, and Eleazar Eskin. 1999. Towards multidocument summarization by reformulation: Progress and prospects. In *In Proceedings of AAAI-99*.

Vivi Nastase. 2008. Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

A. Nenkova and L. Vanderwende. 2005. The impact of frequency on summarization. Technical report, Microsoft Research.

Dragomir R. Radev. 2004. Lexrank: graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR*.

M. Steyvers and T. Griffiths, 2007. *Probabilistic Topic Models*.

Kristina Toutanova, Chris Brockett, Michael Gamon Jagadeesh Jagarlamudi, Hisami Suzuki, and Lucy Vanderwende. 2007. The pythy summarization system: Microsoft research at duc 2007. In *DUC*.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. volume 43.

Xiaojun Wan and Jianwu Yang. 2006. Improved affinity graph based multi-document summarization. In *HLT-NAACL*.

Xuerui Wang, Andrew McCallum, and Xing Wei. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *ICDM*.

Global Models of Document Structure Using Latent Permutations

Harr Chen, S.R.K. Branavan, Regina Barzilay, David R. Karger
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{harr, branavan, regina, karger}@csail.mit.edu

Abstract

We present a novel Bayesian topic model for learning discourse-level document structure. Our model leverages insights from discourse theory to constrain latent topic assignments in a way that reflects the underlying organization of document topics. We propose a *global* model in which both topic selection and ordering are biased to be similar across a collection of related documents. We show that this space of orderings can be elegantly represented using a distribution over permutations called the *generalized Mallows model*. Our structure-aware approach substantially outperforms alternative approaches for cross-document comparison and single-document segmentation.[1]

1 Introduction

In this paper, we introduce a novel latent topic model for the unsupervised learning of document structure. Traditional topic models assume that topics are randomly spread throughout a document, or that the succession of topics in a document is Markovian. In contrast, our approach takes advantage of two important discourse-level properties of text in determining topic assignments: first, that each document follows a progression of nonrecurring coherent topics (Halliday and Hasan, 1976); and second, that documents from the same domain tend to present similar topics, in similar orders (Wray, 2002). We show that a topic model incorporating these long-range dependencies outperforms alternative approaches for segmentation and cross-document comparison.

For example, consider a collection of encyclopedia articles about cities. The first constraint captures the notion that a single topic, such as Architecture, is expressed in a contiguous block within the document, rather than spread over disconnected sections. The second constraint reflects our intuition that all of these related articles will generally mention some major topics associated with cities, such as History and Culture, and will often exhibit similar topic orderings, such as placing History before Culture.

We present a Bayesian latent topic model over related documents that encodes these discourse constraints by positing a single distribution over a document's *entire* topic structure. This global view on ordering is able to elegantly encode discourse-level properties that would be difficult to represent using local dependencies, such as those induced by hidden Markov models. Our model enforces that the same topic does not appear in disconnected portions of the topic sequence. Furthermore, our approach biases toward selecting sequences with similar topic *ordering*, by modeling a distribution over the space of topic permutations.

Learning this ordering distribution is a key technical challenge in our proposed approach. For this purpose, we employ the *generalized Mallows model*, a permutation distribution that concentrates probability mass on a small set of similar permutations. It directly captures the intuition of the second constraint, and uses a small parameter set to control how likely individual topics are to be reordered.

We evaluate our model on two challenging

[1] Code, data, and annotations used in this work are available at http://groups.csail.mit.edu/rbg/code/mallows/

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 371–379,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

document-level tasks. In the *alignment* task, we aim to discover paragraphs across different documents that share the same topic. We also consider the *segmentation* task, where the goal is to partition each document into a sequence of topically coherent segments. We find that our structure modeling approach substantially outperforms state-of-the-art baselines for both tasks. Furthermore, we demonstrate the importance of explicitly modeling a distribution over topic permutations; our model yields significantly better results than variants that either use a fixed ordering, or are order-agnostic.

2 Related Work

Topic and Content Models Our work is grounded in topic modeling approaches, which posit that latent state variables control the generation of words. In earlier topic modeling work such as *latent Dirichlet allocation* (LDA) (Blei et al., 2003; Griffiths and Steyvers, 2004), documents are treated as bags of words, where each word receives a separate topic assignment; the topic assignments are auxiliary variables to the main task of language modeling.

More recent work has attempted to adapt the concepts of topic modeling to more sophisticated representations than a bag of words; they use these representations to impose stronger constraints on topic assignments (Griffiths et al., 2005; Wallach, 2006; Purver et al., 2006; Gruber et al., 2007). These approaches, however, generally model Markovian topic or state transitions, which only capture local dependencies between adjacent words or blocks within a document. For instance, content models (Barzilay and Lee, 2004; Elsner et al., 2007) are implemented as HMMs, where the states correspond to topics of domain-specific information, and transitions reflect pairwise ordering preferences. Even approaches that break text into contiguous chunks (Titov and McDonald, 2008) assign topics based on local context. While these locally constrained models can implicitly reflect some discourse-level constraints, they cannot capture long-range dependencies without an explosion of the parameter space. In contrast, our model captures the entire sequence of topics using a compact representation. As a result, we can explicitly and tractably model global discourse-level constraints.

Modeling Ordering Constraints Sentence ordering has been extensively studied in the context of probabilistic text modeling for summarization and generation (Barzilay et al., 2002; Lapata, 2003; Karamanis et al., 2004). The emphasis of that body of work is on learning ordering constraints from data, with the goal of reordering new text from the same domain. Our emphasis, however, is on applications where ordering is already observed, and how that ordering can improve text analysis. From the methodological side, that body of prior work is largely driven by local pairwise constraints, while we aim to encode global constraints.

3 Problem Formulation

Our document structure learning problem can be formalized as follows. We are given a corpus of D related documents. Each document expresses some subset of a common set of K topics. We assign a single topic to each paragraph,[2] incorporating the notion that paragraphs are internally topically consistent (Halliday and Hasan, 1976). To capture the discourse constraint on topic progression described in Section 1, we require that topic assignments be contiguous within each document.[3] Furthermore, we assume that the underlying topic sequences exhibit similarity across documents. Our goal is to recover a *topic assignment* for each paragraph in the corpus, subject to these constraints.

Our formulation shares some similarity with the standard LDA setup, in that a common set of topics is assigned across a collection of documents. However, in LDA each word's topic assignment is conditionally independent, following the bag of words view of documents. In contrast, our constraints on how topics are assigned let us connect word distributional patterns to document-level topic structure.

4 Model

We propose a generative Bayesian model that explains how a corpus of D documents, given as sequences of paragraphs, can be produced from a set of hidden topic variables. Topic assignments to each

[2] Note that our analysis applies equally to other levels of textual granularity, such as sentences.

[3] That is, if paragraphs i and j are assigned the same topic, every paragraph between them must have that topic.

paragraph, ranging from 1 to K, are the model's final output, implicitly grouping topically similar paragraphs. At a high level, the process first selects the bag of topics to be expressed in the document, and how they are ordered; these topics then determine the selection of words for each paragraph.

For each document d with N_d paragraphs, we separately generate a *bag of topics* $\mathbf{t}_d$ and a *topic ordering* π_d. The unordered bag of topics, which contains N_d elements, expresses how many paragraphs of the document are assigned to each of the K topics. Note that some topics may not appear at all. Variable $\mathbf{t}_d$ is constructed by taking N_d samples from a distribution over topics τ, a multinomial representing the probability of each topic being expressed. Sharing τ between documents captures the intuition that certain topics are more likely across the entire corpus.

The topic ordering variable π_d is a permutation over the numbers 1 through K that defines the order in which topics appear in the document. We draw π_d from the *generalized Mallows model*, a distribution over permutations that we explain in Section 4.1. As we will see, this particular distribution biases the permutation selection to be close to a single centroid, reflecting the discourse constraint of preferring similar topic structures across documents.

Together, a document's bag of topics $\mathbf{t}_d$ and ordering π_d determine the topic assignment $z_{d,p}$ for each of its paragraphs. For example, in a corpus with $K = 4$, a seven-paragraph document d with $\mathbf{t}_d = \{1, 1, 1, 1, 2, 4, 4\}$ and $\pi_d = (2\ 4\ 3\ 1)$ would induce the topic sequence $\mathbf{z}_d = (2\ 4\ 4\ 1\ 1\ 1\ 1)$. The induced topic sequence $\mathbf{z}_d$ can never assign the same topic to two unconnected portions of a document, thus satisfying the constraint of topic contiguity.

As with LDA, we assume that each topic k is associated with a language model θ_k. The words of a paragraph assigned to topic k are then drawn from that topic's language model θ_k.

Before turning to a more formal discussion of the generative process, we first provide background on the permutation model for topic ordering.

4.1 The Generalized Mallows Model

A central challenge of the approach we take is modeling the distribution over possible topic permutations. For this purpose we use the generalized Mallows model (GMM) (Fligner and Verducci, 1986; Lebanon and Lafferty, 2002; Meilă et al., 2007), which exhibits two appealing properties in the context of this task. First, the model concentrates probability mass on some "canonical" ordering and small perturbations of that ordering. This characteristic matches our constraint that documents from the same domain exhibit structural similarity. Second, its parameter set scales linearly with the permutation length, making it sufficiently constrained and tractable for inference. In general, this distribution could potentially be applied to other NLP applications where ordering is important.

Permutation Representation Typically, permutations are represented directly as an ordered sequence of elements. The GMM utilizes an alternative representation defined as a vector $(v_1, \ldots, v_{K-1})$ of *inversion counts* with respect to the identity permutation $(1, \ldots, K)$. Term v_j counts the number of times a value greater than j appears before j in the permutation.[4] For instance, given the standard-form permutation $(3\ 1\ 5\ 2\ 4)$, $v_2 = 2$ because 3 and 5 appear before 2; the entire inversion count vector would be $(1\ 2\ 0\ 1)$. Every vector of inversion counts uniquely identifies a single permutation.

The Distribution The GMM assigns probability mass according to the distance of a given permutation from the identity permutation $\{1, \ldots, K\}$, based on $K - 1$ real-valued parameters $(\rho_1, \ldots \rho_{K-1})$.[5] Using the inversion count representation of a permutation, the GMM's probability mass function is expressed as an independent product of probabilities for each v_j:

$$\begin{aligned}\text{GMM}(\mathbf{v} \mid \rho) &= \frac{e^{-\sum_j \rho_j v_j}}{\psi(\rho)} \\ &= \prod_{j=1}^{n-1} \frac{e^{-\rho_j v_j}}{\psi_j(\rho_j)}, \end{aligned} \quad (1)$$

where $\psi_j(\rho_j)$ is a normalization factor with value:

$$\psi_j(\rho_j) = \frac{1 - e^{-(K-j+1)\rho_j}}{1 - e^{-\rho_j}}.$$

[4] The sum of a vector of inversion counts is simply that permutation's Kendall's τ distance to the identity permutation.

[5] In our work we take the identity permutation to be the fixed centroid, which is a parameter in the full GMM. As we explain later, our model is not hampered by this apparent restriction.

Due to the exponential form of the distribution, requiring that $\rho_j > 0$ constrains the GMM to assign highest probability mass to each v_j being zero, corresponding to the identity permutation. A higher value for ρ_j assigns more probability mass to v_j being close to zero, biasing j to have fewer inversions.

The GMM elegantly captures our earlier requirement for a probability distribution that concentrates mass around a global ordering, and uses few parameters to do so. Because the topic numbers in our task are completely symmetric and not linked to any extrinsic observations, fixing the identity permutation to be that global ordering does not sacrifice any representational power. Another major benefit of the GMM is its membership in the exponential family of distributions; this means that it is particularly amenable to a Bayesian representation, as it admits a natural conjugate prior:

$$\text{GMM}_0(\rho_j \mid v_{j,0}, \nu_0) \propto e^{(-\rho_j v_{j,0} - \log \psi_j(\rho_j))\nu_0}. \quad (2)$$

Intuitively, this prior states that over ν_0 prior trials, the total number of inversions was $\nu_0 v_{j,0}$. This distribution can be easily updated with the observed v_j to derive a posterior distribution.[6]

4.2 Formal Generative Process

We now fully specify the details of our model. We observe a corpus of D documents, each an ordered sequence of paragraphs, and a specification of a number of topics K. Each paragraph is represented as a bag of words. The model induces a set of hidden variables that probabilistically explain how the words of the corpus were produced. Our final desired output is the distributions over the paragraphs' hidden topic assignment variables. In the following, variables subscripted with 0 are fixed prior hyperparameters.

1. For each topic k, draw a language model $\theta_k \sim \text{Dirichlet}(\theta_0)$. As with LDA, these are topic-specific word distributions.
2. Draw a topic distribution $\tau \sim \text{Dirichlet}(\tau_0)$, which expresses how likely each topic is to appear regardless of position.
3. Draw the topic ordering distribution parameters $\rho_j \sim \text{GMM}_0(\rho_0, \nu_0)$ for $j = 1$ to $K - 1$. These parameters control how rapidly probability mass decays for having more inversions for each topic. A separate ρ_j for every topic allows us to learn that some topics are more likely to be reordered than others.
4. For each document d with N_d paragraphs:
   (a) Draw a bag of topics $\mathbf{t}_d$ by sampling N_d times from Multinomial(τ).
   (b) Draw a topic ordering π_d by sampling a vector of inversion counts $\mathbf{v}_d \sim \text{GMM}(\rho)$.
   (c) Compute the vector of topic assignments $\mathbf{z}_d$ for document d's paragraphs, by sorting $\mathbf{t}_d$ according to π_d.[7]
   (d) For each paragraph p in document d:
      i. Sample each word $w_{d,p,j}$ according to the language model of p: $w_{d,p,j} \sim \text{Multinomial}(\theta_{z_{d,p}})$.

5 Inference

The variables that we aim to infer are the topic assignments z of each paragraph, which are determined by the bag of topics $\mathbf{t}$ and ordering π for each document. Thus, our goal is to estimate the marginal distributions of $\mathbf{t}$ and π given the document text.

We accomplish this inference task through Gibbs sampling (Bishop, 2006). A Gibbs sampler builds a Markov chain over the hidden variable state space whose stationary distribution is the actual posterior of the joint distribution. Each new sample is drawn from the distribution of a single variable conditioned on previous samples of the other variables. We can "collapse" the sampler by integrating over some of the hidden variables in the model, in effect reducing the state space of the Markov chain. Collapsed sampling has been previously demonstrated to be effective for LDA and its variants (Griffiths and Steyvers, 2004; Porteous et al., 2008; Titov and McDonald, 2008). Our sampler integrates over all but three sets

[6] Because each v_j has a different range, it is inconvenient to set the prior hyperparameters $v_{j,0}$ directly. In our work, we instead fix the mode of the prior distribution to a value ρ_0, which works out to setting $v_{j,0} = \frac{1}{\exp(\rho_0)-1} - \frac{K-j+1}{\exp((K-j+1)\rho_0)-1}$.

[7] Multiple permutations can contribute to the probability of a single document's topic assignments $\mathbf{z}_d$, if there are topics that do not appear in $\mathbf{t}_d$. As a result, our current formulation is biased toward assignments with fewer topics per document. In practice, we do not find this to negatively impact model performance.

of hidden variables: bags of topics $\mathbf{t}$, orderings π, and permutation inversion parameters ρ. After a burn-in period, we treat the last samples of $\mathbf{t}$ and π as a draw from the true posterior.

Document Probability As a preliminary step, consider how to calculate the probability of a single document's words $\mathbf{w}_d$ given the document's paragraph topic assignments $\mathbf{z}_d$, and other documents and their topic assignments. Note that this probability is decomposable into a product of probabilities over individual paragraphs, where paragraphs with different topics have conditionally independent word probabilities. Let $\mathbf{w}_{-d}$ and $\mathbf{z}_{-d}$ indicate the words and topic assignments to documents other than d, and W be the vocabulary size. The probability of the words in d is then:

$$\begin{aligned}
&P(\mathbf{w}_d \mid \mathbf{z}, \mathbf{w}_{-d}, \theta_0) \\
&= \prod_{k=1}^{K} \int_{\theta_k} P(\mathbf{w}_d \mid \mathbf{z}_d, \theta_k) P(\theta_k \mid \mathbf{z}, \mathbf{w}_{-d}, \theta_0) d\theta_k \\
&= \prod_{k=1}^{K} \text{DCM}(\{\mathbf{w}_{d,i} : z_{d,i} = k\} \\
&\qquad \mid \{\mathbf{w}_{-d,i} : z_{-d,i} = k\}, \theta_0), \qquad (3)
\end{aligned}$$

where DCM$(\cdot)$ refers to the *Dirichlet compound multinomial* distribution, the result of integrating over multinomial parameters with a Dirichlet prior (Bernardo and Smith, 2000). For a Dirichlet prior with parameters $\alpha = (\alpha_1, \ldots, \alpha_W)$, the DCM assigns the following probability to a series of observations $\mathbf{x} = \{x_1, \ldots, x_n\}$:

$$\text{DCM}(\mathbf{x} \mid \alpha) = \frac{\Gamma(\sum_j \alpha_j)}{\prod_j \Gamma(\alpha_j)} \prod_{i=1}^{W} \frac{\Gamma(N(\mathbf{x}, i) + \alpha_i)}{\Gamma(|\mathbf{x}| + \sum_j \alpha_j)},$$

where $N(\mathbf{x}, i)$ refers to the number of times word i appears in $\mathbf{x}$. Here, $\Gamma(\cdot)$ is the Gamma function, a generalization of the factorial for real numbers. Some algebra shows that the DCM's posterior probability density function conditioned on a series of observations $\mathbf{y} = \{y_1, \ldots, y_n\}$ can be computed by updating each α_i with counts of how often word i appears in $\mathbf{y}$:

$$\begin{aligned}
&\text{DCM}(\mathbf{x} \mid \mathbf{y}, \alpha) \\
&= \text{DCM}(\mathbf{x} \mid \alpha_1 + N(\mathbf{y}, 1), \ldots, \alpha_W + N(\mathbf{y}, W)). \qquad (4)
\end{aligned}$$

Equation 3 and 4 will be used again to compute the conditional distributions of the hidden variables.

We now turn to a discussion of how each individual random variable is resampled.

Bag of Topics First we consider how to resample $t_{d,i}$, the ith topic draw for document d conditioned on all other parameters being fixed (note this is *not* the topic of the ith paragraph, as we reorder topics using π_d):

$$\begin{aligned}
&P(t_{d,i} = t \mid \ldots) \\
&\propto P(t_{d,i} = t \mid \mathbf{t}_{-(d,i)}, \tau_0) P(\mathbf{w}_d \mid \mathbf{t}_d, \pi_d, \mathbf{w}_{-d}, \mathbf{z}_{-d}, \theta_0) \\
&\propto \frac{N(\mathbf{t}_{-(d,i)}, t) + \tau_0}{|\mathbf{t}_{-(d,i)}| + K\tau_0} P(\mathbf{w}_d \mid \mathbf{z}, \mathbf{w}_{-d}, \theta_0),
\end{aligned}$$

where $\mathbf{t}_d$ is updated to reflect $t_{d,i} = t$, and $\mathbf{z}_d$ is deterministically computed by mapping $\mathbf{t}_d$ and π_d to actual paragraph topic assignments. The first step reflects an application of Bayes rule to factor out the term for $\mathbf{w}_d$. In the second step, the first term arises out of the DCM, by updating the parameters τ_0 with observations $\mathbf{t}_{-(d,i)}$ as in equation 4 and dropping constants. The document probability term is computed using equation 3. The new $t_{d,i}$ is selected by sampling from this probability computed over all possible topic assignments.

Ordering The parameterization of a permutation π as a series of inversion values v_j reveals a natural way to decompose the search space for Gibbs sampling. For a single ordering, each v_j can be sampled independently, according to:

$$\begin{aligned}
&P(v_j = v \mid \ldots) \\
&\propto P(v_j = v \mid \rho_j) P(\mathbf{w}_d \mid \mathbf{t}_d, \pi_d, \mathbf{w}_{-d}, \mathbf{z}_{-d}, \theta_0) \\
&= \text{GMM}_j(v \mid \rho_j) P(\mathbf{w}_d \mid \mathbf{z}_d, \mathbf{w}_{-d}, \mathbf{z}_{-d}, \theta_0),
\end{aligned}$$

where π_d is updated to reflect $v_j = v$, and $\mathbf{z}_d$ is computed according to $\mathbf{t}_d$ and π_d. The first term refers to the jth multiplicand of equation 1; the second is computed using equation 3. Term v_j is sampled according to the resulting probabilities.

GMM Parameters For each $j = 1$ to $K - 1$, we resample ρ_j from its posterior distribution:

$$\begin{aligned}
&P(\rho_j \mid \ldots) \\
&= \text{GMM}_0 \left(\rho_j \,\middle|\, \frac{\sum_i v_{j,i} + v_{j,0}\nu_0}{N + \nu_0}, N + \nu_0 \right),
\end{aligned}$$

where GMM_0 is evaluated according to equation 2. The normalization constant of this distribution is unknown, meaning that we cannot directly compute and invert the cumulative distribution function to sample from this distribution. However, the distribution itself is univariate and unimodal, so we can expect that an MCMC technique such as *slice sampling* (Neal, 2003) should perform well. In practice, the MATLAB black-box slice sampler provides a robust draw from this distribution.

6 Experimental Setup

Data Sets We evaluate our model on two data sets drawn from the English Wikipedia. The first set is 100 articles about large cities, with topics such as History, Culture, and Demographics. The second is 118 articles about chemical elements in the periodic table, including topics such as Biological Role, Occurrence, and Isotopes. Within each corpus, articles often exhibit similar section orderings, but many have idiosyncratic inversions. This structural variability arises out of the collaborative nature of Wikipedia, which allows articles to evolve independently. Corpus statistics are summarized below.

Corpus	Docs	Paragraphs	Vocab	Words
Cities	100	6,670	41,978	492,402
Elements	118	2,810	18,008	191,762

In each data set, the articles' *noisy section headings* induce a reference structure to compare against. This reference structure assumes that two paragraphs are aligned if and only if their section headings are identical, and that section boundaries provide the correct segmentation of each document. These headings are only used for evaluation, and are not provided to any of the systems.

Using the section headings to build the reference structure can be problematic, as the same topic may be referred to using different titles across different documents, and sections may be divided at differing levels of granularity. Thus, for the Cities data set, we manually annotated each article's paragraphs with a consistent set of section headings, providing us an additional reference structure to evaluate against. In this *clean section headings* set, we found approximately 18 topics that were expressed in more than one document.

Tasks and Metrics We study performance on the tasks of alignment and segmentation. In the former task, we measure whether paragraphs identified to be the same topic by our model have the same section headings, and vice versa. First, we identify the "closest" topic to each section heading, by finding the topic that is most commonly assigned to paragraphs under that section heading. We compute the proportion of paragraphs where the model's topic assignment matches the section heading's topic, giving us a *recall* score. High recall indicates that paragraphs of the same section headings are always being assigned to the same topic. Conversely, we can find the closest section heading to each topic, by finding the section heading that is most common for the paragraphs assigned to a single topic. We then compute the proportion of paragraphs from that topic whose section heading is the same as the reference heading for that topic, yielding a *precision* score. High precision means that paragraphs assigned to a single topic usually correspond to the same section heading. The harmonic mean of recall and precision is the summary *F-score*.

Statistical significance in this setup is measured with *approximate randomization* (Noreen, 1989), a nonparametric test that can be directly applied to nonlinear metrics such as F-score. This test has been used in prior evaluations for information extraction and machine translation (Chinchor, 1995; Riezler and Maxwell, 2005).

For the second task, we take the boundaries at which topics change within a document to be a segmentation of that document. We evaluate using the standard penalty metrics P_k and WindowDiff (Beeferman et al., 1999; Pevzner and Hearst, 2002). Both pass a sliding window over the documents and compute the probability of the words at the ends of the windows being improperly segmented with respect to each other. WindowDiff requires that the number of segmentation boundaries between the endpoints be correct as well.[8]

Our model takes a parameter K which controls the upper bound on the number of latent topics. Note that our algorithm can select fewer than K topics for each document, so K does not determine the number

[8] Statistical significance testing is not standardized and usually not reported for the segmentation task, so we omit these tests in our results.

of segments in each document. We report results using both $K = 10$ and 20 (recall that the cleanly annotated Cities data set had 18 topics).

Baselines and Model Variants We consider baselines from the literature that perform either alignment or segmentation. For the first task, we compare against the *hidden topic Markov model* (HTMM) (Gruber et al., 2007), which represents topic transitions between adjacent paragraphs in a Markovian fashion, similar to the approach taken in content modeling work. Note that HTMM can only capture local constraints, so it would allow topics to recur noncontiguously throughout a document.

We also compare against the structure-agnostic approach of clustering the paragraphs using the CLUTO toolkit,[9] which uses repeated bisection to maximize a cosine similarity-based objective.

For the segmentation task, we compare to BayesSeg (Eisenstein and Barzilay, 2008),[10] a Bayesian topic-based segmentation model that outperforms previous segmentation approaches (Utiyama and Isahara, 2001; Galley et al., 2003; Purver et al., 2006; Malioutov and Barzilay, 2006). BayesSeg enforces the topic contiguity constraint that motivated our model. We provide this baseline with the benefit of knowing the correct number of segments for each document, which is not provided to our system. Note that BayesSeg processes each document individually, so it cannot capture structural relatedness across documents.

To investigate the importance of our ordering model, we consider two variants of our model that alternately relax and tighten ordering constraints. In the *constrained* model, we require all documents to follow the same canonical ordering of topics. This is equivalent to forcing the topic permutation distribution to give all its probability to one ordering, and can be implemented by fixing all inversion counts $\mathbf{v}$ to zero during inference. At the other extreme, we consider the *uniform* model, which assumes a uniform distribution over all topic permutations instead of biasing toward a small related set. In our implementation, this can be simulated by forcing the GMM parameters ρ to always be zero. Both variants still enforce topic contiguity, and allow segments across documents to be aligned by topic assignment.

Evaluation Procedures For each evaluation of our model and its variants, we run the Gibbs sampler from five random seed states, and take the 10,000th iteration of each chain as a sample. Results shown are the average over these five samples. All Dirichlet prior hyperparameters are set to 0.1, encouraging sparse distributions. For the GMM, we set the prior decay parameter ρ_0 to 1, and the sample size prior ν_0 to be 0.1 times the number of documents.

For the baselines, we use implementations publicly released by their authors. We set HTMM's priors according to values recommended in the authors' original work. For BayesSeg, we use its built-in hyperparameter re-estimation mechanism.

7 Results

Alignment Table 1 presents the results of the alignment evaluation. In every case, the best performance is achieved using our full model, by a statistically significant and usually substantial margin.

In both domains, the baseline clustering method performs competitively, indicating that word cues alone are a good indicator of topic. While the simpler variations of our model achieve reasonable performance, adding the richer GMM distribution consistently yields superior results.

Across each of our evaluations, HTMM greatly underperforms the other approaches. Manual examination of the actual topic assignments reveals that HTMM often selects the same topic for disconnected paragraphs of the same document, violating the topic contiguity constraint, and demonstrating the importance of modeling global constraints for document structure tasks.

We also compare performance measured on the manually annotated section headings against the actual noisy headings. The ranking of methods by performance remains mostly unchanged between these two evaluations, indicating that the noisy headings are sufficient for gaining insight into the comparative performance of the different approaches.

Segmentation Table 2 presents the segmentation experiment results. On both data sets, our model

[9] http://glaros.dtc.umn.edu/gkhome/views/cluto/

[10] We do not evaluate on the corpora used in their work, since our model relies on content similarity across documents in the corpus.

		Cities: clean headings			Cities: noisy headings			Elements: noisy headings		
		Recall	Prec	F-score	Recall	Prec	F-score	Recall	Prec	F-score
$K = 10$	Clustering	0.578	0.439	$*$ 0.499	0.611	0.331	$*$ 0.429	0.524	0.361	$*$ 0.428
	HTMM	0.446	0.232	$*$ 0.305	0.480	0.183	$*$ 0.265	0.430	0.190	$*$ 0.264
	Constrained	0.579	0.471	$*$ 0.520	0.667	0.382	$*$ 0.485	0.603	0.408	$*$ 0.487
	Uniform	0.520	0.440	$*$ 0.477	0.599	0.343	$*$ 0.436	0.591	0.403	$*$ 0.479
	Our model	**0.639**	**0.509**	**0.566**	**0.705**	**0.399**	**0.510**	**0.685**	**0.460**	**0.551**
$K = 20$	Clustering	0.486	0.541	$*$ 0.512	0.527	0.414	$*$ 0.464	0.477	0.402	$*$ 0.436
	HTMM	0.260	0.217	$*$ 0.237	0.304	0.187	$*$ 0.232	0.248	0.243	$*$ 0.246
	Constrained	0.458	0.519	$*$ 0.486	0.553	0.415	$*$ 0.474	0.510	0.421	$*$ 0.461
	Uniform	0.499	0.551	$*$ 0.524	0.571	0.423	$*$ 0.486	0.550	0.479	$\diamond$ 0.512
	Our model	**0.578**	**0.636**	**0.606**	**0.648**	**0.489**	**0.557**	**0.569**	**0.498**	**0.531**

Table 1: Comparison of the alignments produced by our model and a series of baselines and model variations, for both 10 and 20 topics, evaluated against clean and noisy sets of section headings. Higher scores are better. Within the same K, the methods which our model significantly outperforms are indicated with $*$ for $p < 0.001$ and $\diamond$ for $p < 0.01$.

		Cities: clean headings			Cities: noisy headings			Elements: noisy headings		
		P_k	WD	# Segs	P_k	WD	# Segs	P_k	WD	# Segs
BayesSeg		0.321	0.376	† 12.3	0.317	0.376	† 13.2	0.279	0.316	† 7.7
$K = 10$	Constrained	0.260	**0.281**	7.7	0.267	0.288	7.7	0.227	0.244	5.4
	Uniform	0.268	0.300	8.8	0.273	0.304	8.8	0.226	0.250	6.6
	Our model	**0.253**	0.283	9.0	**0.257**	**0.286**	9.0	**0.201**	**0.226**	6.7
$K = 20$	Constrained	0.274	0.314	10.9	0.274	0.313	10.9	0.231	0.257	6.6
	Uniform	0.234	0.294	14.0	0.234	0.290	14.0	0.209	0.248	8.7
	Our model	**0.221**	**0.278**	14.2	**0.222**	**0.278**	14.2	**0.203**	**0.243**	8.6

Table 2: Comparison of the segmentations produced by our model and a series of baselines and model variations, for both 10 and 20 topics, evaluated against clean and noisy sets of section headings. Lower scores are better. †BayesSeg is given the true number of segments, so its segments count reflects the reference structure's segmentation.

outperforms the BayesSeg baseline by a substantial margin regardless of K. This result provides strong evidence that learning connected topic models over related documents leads to improved segmentation performance. In effect, our model can take advantage of shared structure across related documents.

In all but one case, the best performance is obtained by the full version of our model. This result indicates that enforcing discourse-motivated structural constraints allows for better segmentation induction. Encoding global discourse-level constraints leads to better language models, resulting in more accurate predictions of segment boundaries.

8 Conclusions

In this paper, we have shown how an unsupervised topic-based approach can capture document structure. Our resulting model constrains topic assignments in a way that requires global modeling of entire topic sequences. We showed that the generalized Mallows model is a theoretically and empirically appealing way of capturing the ordering component of this topic sequence. Our results demonstrate the importance of augmenting statistical models of text analysis with structural constraints motivated by discourse theory.

Acknowledgments

The authors acknowledge the funding support of NSF CAREER grant IIS-0448168, the NSF Graduate Fellowship, the Office of Naval Research, Quanta, Nokia, and the Microsoft Faculty Fellowship. We thank the members of the NLP group at MIT and numerous others who offered suggestions and comments on this work. We are especially grateful to Marina Meilă for introducing us to the Mallows model. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors, and do not necessarily reflect the views of the funding organizations.

References

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of NAACL/HLT*.

Regina Barzilay, Noemie Elhadad, and Kathleen McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17:35–55.

Doug Beeferman, Adam Berger, and John D. Lafferty. 1999. Statistical models for text segmentation. *Machine Learning*, 34:177–210.

José M. Bernardo and Adrian F.M. Smith. 2000. *Bayesian Theory*. Wiley Series in Probability and Statistics.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Nancy Chinchor. 1995. Statistical significance of MUC-6 results. In *Proceedings of the 6th Conference on Message Understanding*.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of EMNLP*.

Micha Elsner, Joseph Austerweil, and Eugene Charniak. 2007. A unified local and global model for discourse coherence. In *Proceedings of NAACL/HLT*.

M.A. Fligner and J.S. Verducci. 1986. Distance based ranking models. *Journal of the Royal Statistical Society, Series B*, 48(3):359–369.

Michel Galley, Kathleen R. McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of ACL*.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235.

Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. 2005. Integrating topics and syntax. In *Advances in NIPS*.

Amit Gruber, Michal Rosen-Zvi, and Yair Weiss. 2007. Hidden topic markov models. In *Proceedings of AISTATS*.

M. A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman.

Nikiforos Karamanis, Massimo Poesio, Chris Mellish, and Jon Oberlander. 2004. Evaluating centering-based metrics of coherence for text structuring using a reliably annotated corpus. In *Proceedings of ACL*.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of ACL*.

Guy Lebanon and John Lafferty. 2002. Cranking: combining rankings using conditional probability models on permutations. In *Proceedings of ICML*.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of ACL*.

Marina Meilă, Kapil Phadnis, Arthur Patterson, and Jeff Bilmes. 2007. Consensus ranking under the exponential model. In *Proceedings of UAI*.

Radford M. Neal. 2003. Slice sampling. *Annals of Statistics*, 31:705–767.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses. An Introduction*. Wiley.

Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28:19–36.

Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. 2008. Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of SIGKDD*.

Matthew Purver, Konrad Körding, Thomas L. Griffiths, and Joshua B. Tenenbaum. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *Proceedings of ACL/COLING*.

Stefan Riezler and John T. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

Ivan Titov and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *Proceedings of WWW*.

Masao Utiyama and Hitoshi Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of ACL*.

Hanna M. Wallach. 2006. Topic modeling: beyond bag of words. In *Proceedings of ICML*.

Alison Wray. 2002. *Formulaic Language and the Lexicon*. Cambridge University Press, Cambridge.

Assessing and Improving the Performance of Speech Recognition for Incremental Systems

Timo Baumann, Michaela Atterer, David Schlangen
Institut für Linguistik
Universität Potsdam
Potsdam, Germany
{timo,atterer,das}@ling.uni-potsdam.de

Abstract

In incremental spoken dialogue systems, partial hypotheses about what was said are required even while the utterance is still ongoing. We define measures for evaluating the quality of incremental ASR components with respect to the *relative correctness* of the partial hypotheses compared to hypotheses that can optimize over the complete input, the *timing* of hypothesis formation relative to the portion of the input they are about, and hypothesis *stability*, defined as the number of times they are revised. We show that simple incremental post-processing can improve stability dramatically, at the cost of timeliness (from 90 % of edits of hypotheses being spurious down to 10 % at a lag of 320 ms). The measures are not independent, and we show how system designers can find a desired operating point for their ASR. To our knowledge, we are the first to suggest and examine a variety of measures for assessing incremental ASR and improve performance on this basis.

1 Introduction

Incrementality, that is, the property of beginning to process input before it is complete, is often seen as a desirable property of dialogue systems (e.g., Allen et al. (2001)), as it allows the system to (a) *fold* processing time (of modules such as parsers, or dialogue managers) into the time taken by the utterance, and (b) *react* to partial results, for example by generating back-channel utterances or speculatively initiating potentially relevant database queries.

Input to a spoken dialogue system normally passes an automatic speech recognizer (ASR) as a first processing module, thus the module's incrementality determines the level of incrementality that can be reached by the system as a whole. Using an ASR system incrementally poses interesting challenges, however. Typically, ASRs use dynamic programming and the maximum likelihood hypothesis to find the word sequence with the lowest expected likelihood of the sequence containing errors (sentence error). Due to the dynamic programming approach, what is considered the best hypothesis about a given stretch of the input signal can change during the recognition process, as more right context which can be used as evidence becomes available.

In this paper, we argue that normally used metrics for ASR evaluation such as word error rate must be complemented with metrics specifically designed for measuring incremental performance, and offer some such metrics. We show that there are various subproperties that are not independent of each other, and that trade-offs are involved if either of those is to be optimized. Finally, we propose ways to improve incremental performance (as measured by our metrics) through the use of smoothing techniques.

To our knowledge, incremental evaluation metrics of ASR for incremental systems have not yet been covered in the literature. Most closely related, Wachsmuth et al. (1998) show results for an ASR which fixes its results after a given time Δ and report the corresponding word error rate (WER). This unfortunately confounds the incremental and non-incremental properties of their ASR's performance.

The remainder of this paper is structured as follows: In section 2, we give an overview of incremenality with respect to ASR, and develop our evalua-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 380–388,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

tion metrics. Section 3 describes the setup and data that we used in our experiments, and reports and discusses some basic measures for different variants of the setup. In section 4 we propose and discuss two orthogonal methods that improve incremental performance: using right context and using message smoothing, which show different properties with regard to our measures. Finally, in section 5 we sum up and point to future directions.

2 Incrementality and Evaluation Measures for Incremental ASR

In a modular system, an *incremental module* is one that generates (partial) responses while input is still ongoing and makes these available to other modules (Kilger and Finkler, 1995). ASR modules that use token passing (Young et al., 1989) can easily be adapted to output a new, *live* hypothesis after processing of every input frame (often that is every 10 ms). In an incremental system we are able to get partial results from these hypotheses as soon as they become available – or rather as soon as they can be trusted. As mentioned above, hypotheses are only tentative, and may be revised when more right context becomes available. Modules consuming the output of an incremental ASR hence must be able to deal with such revisions. There is a first trade-off here: Depending on how costly revision is for later modules (which after all may need to revise any hypotheses which they themselves based on the now-revised input), it may be better to reduce the incrementality a bit – in the sense that partial information is produced less often, and hence new words for example are recognised later – if that buys stability (fewer revisions). Also, ignoring some incremental results that are likely to be wrong may increase system performance. Defining these notions more precisely is the aim of this section.

2.1 Relative Correctness

We define a hypothesis at time t (hyp_t) as consisting of a sequence w_{hyp_t} of words predicted by the ASR at time t.[1] As an example figure 1 shows

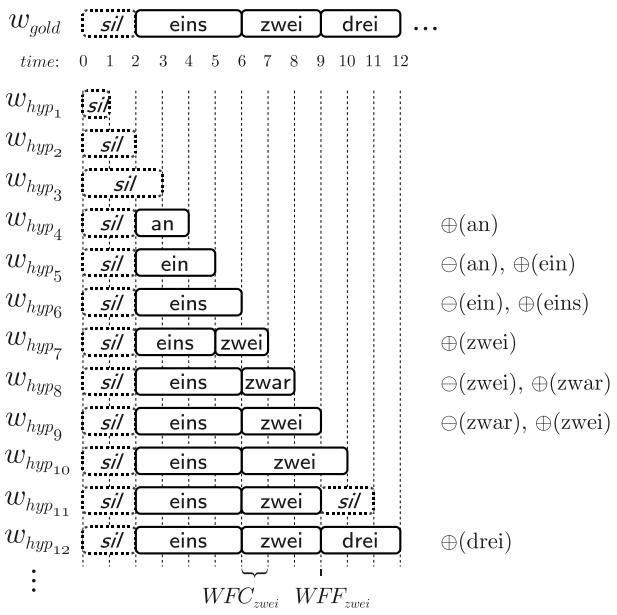


Figure 1: *Live* ASR hypotheses during incremental recognition. Edit messages (see section 2.2) are shown on the right when words are added (⊕) or revoked (⊖). For the word "zwei" WFC and WFF (see section 2.3) are shown at the bottom.

a sequence of incrementally produced hypotheses. (Note that this is an artificial example, showing only a few illustratory and interesting hypotheses. In a real recognition system, the hypothesis frequency is of course much higher, with much repetition of similar hypotheses at consecutive frames.)

The question now is how we can evaluate the quality of a hypothesis at the time t it is produced. It is reasonable to only expect this hypothesis to say something (correct or not) about the input up to time t – unless we want the ASR to *predict*, in which case we want it to make assumptions about times beyond t (see section 4.1). There are two candidates for the yardstick against which the partial hypotheses could be compared: First, one could take the *actually* spoken words, computing measures such as word error rate. The other option, which is the one taken here, is to take as the gold standard the final hypothesis produced by the ASR when it has all evidence avail-

[1] In this paper, we only deal with one-best ASR. We believe that there are no principled differences when generalising to n-best hypotheses, but will explore this in detail in future work.

We also abstract away from changes in the hypothesised start and end times of the words in the sequence. It often happens that the ASR's assumptions about the position of the word boundaries change, even if the word sequence stays constant. If, as we assume here, later modules do not use this timing information, we can consider two hypotheses that only differ in boundary placement as identical.

able (i.e., when the utterance is complete). This is more meaningful for our purpose, as it relates the informativity of the partial hypothesis to what can be expected if the ASR can do all its internal optimisations, and not to the factually correct sequence that the ASR might not be able to recognise even with all information present. This latter problem is already captured in the conventional non-incremental performance measures.

In our metrics in this paper, we hence take as gold standard (w_{gold}) the final, non-incremental hypothesis of the ASR (which, to reiterate this point, might be factually incorrect, that is, might contain word errors). We define a module's incremental response at time t (w_{hyp_t}) as *relatively correct* (*r-correct*), iff it is equal to the non-incremental hypothesis up to time t: $w_{hyp_t t} = w_{gold_t}$. Hence, in figure 1 above, hypotheses 1, 2, 6, 7, 9 and 12 are r-correct.[2] We call the normalised rate of r-correct responses of a module its (average) *r-correctness*.

As defined above, the criterion for r-correctness is still pretty strict, as it demands of the ASR that words on the right edge are recognised even from the first frame on. For example, $w_{hyp_{10}}$ in figure 1 is not r-correct, because $w_{gold_{10}}$ (that part of w_{gold} that ends where $w_{hyp_{10}}$ ends) already spans parts of the word "drei" which has not yet been picked up by the incremental recognition. A relaxed notion of correctness hence is *prefix-correctness*, which requires only that w_{hyp_t} be a prefix of w_{gold_t}. (Hypotheses 3 and 10 in figure 1 are p-correct, as are all r-correct hypotheses.) It should be noted though that p-correctness is too forgiving to be used directly as an optimization target: in the example in figure 1, a module that only ever produces empty hypotheses would trivally achieve perfect p-correctness (as this is always a prefix of w_{gold}).

2.2 Edit Overhead

The measures defined so far capture only static aspects of the incremental performance of a module and do not say anything about the dynamics of the recognition process. To capture this, we look at the changes between subsequent partial hypotheses. There are three ways in which an hypothesis hyp_{t+1} can be different from hyp_t: there can be an *extension* of the word sequence, a *revokation*, or a *revision* of the last words in the sequence.[3] These differences can be expressed as *edit messages*, where extending a sequence by one word would require an *add* message ($\oplus$), deleting the last word in the sequence a *revoke* message ($\ominus$), and exchange of the last word would require two messages, one to revoke the old and one to add the new word.[4]

Now, an incrementally perfect ASR would only generate extensions, adding new words at the right edge; thus, there would be exactly as many edit messages as there are words in w_{gold}. In reality, there are typically many more changes, and hence many spurious edits (see below for characteristic rates in our data). We call the rate of spurious edits the *edit overhead* (EO). For figure 1 above, this is $\frac{8}{11}$: There are 11 edits (as shown in the figure), while we'd expect only 3 (one $\oplus$ for each word in the final result). Hence, 8 edits are spurious.

This measure corresponds directly to the amount of unnecessary activity a consumer of the ASR's output performs when it reacts swiftly to words that may be revoked later on. If the consumer is able to robustly cope with parallel hypotheses (for example by building a lattice-like structure), a high EO may not be problematic, but if revisions are costly for later modules (or even impossible because action has already been taken), we would like EO to be as low as possible. This can be achieved by not sending edit messages unconditionally as soon as words change in the ASR's current hypothesis, using strategies as outlined in section 4. Obviously, deferring or suppressing messages results in delays, a topic to which we turn in the following section, where we define measures for the response time of ASR.

2.3 Timing Measures

So far, our measures capture characteristics about the complete recognition process. We now turn to the timing of the recognition of individual words. For this, we again take the output of the ASR when all signal is present (i.e., w_{gold}) as the basis. There

[2] The timing in hypothesis 7 is not correct – but this does not matter to our notion of correctness (see footnote 1).

[3] As fourth and most frequent alternative, consecutive hypotheses do not change at all.

[4] Revision could also be seen as a third atomic operation, as in standard ASR evaluation (then called "substitution"). To keep things simple, we only regard two atomic operations.

are two things we may be interested in. First, we may want to know when is the first time that a certain word appears in the correct position in the sequence (or equivalently, when its first correct *add* edit message is sent), expressed in relation to its boundaries in w_{gold}. We measure this event, the first time that the ASR was right about a word, relative to its gold beginning. We call the measure *word first correct response* (WFC). As a concrete example take hyp_7 in figure 1. At this point, the word "zwei" is first hypothesised. Compared to the beginning of the word in w_{gold}, this point (t_7) has a delay of 1 frame (the frames are illustrated by the dashed lines).

As explained above, it may very well be the case that for a brief while another hypothesis, not r-correct w.r.t. w_{gold}, may be favoured (cf. the word "zwar" in the example in the figure). Another measure we hence might also be interested in is when our word hypothesis starts remaining stable or, in other words, becomes final. We measure this event relative to the end of the word in the gold standard. We call it *word first final response* (WFF). In our example, again for "zwei", this is t_9, which has a distance of 0 to the right boundary of the word in w_{gold}.

In principle, we could use both anchor points (the left vs. the right edge of a word) for either measure or use a word-relative scale, but for simplicity's sake we restrict ourselves to one anchor point each.

Under normal conditions, we expect WFC to be positive. The better the incremental ASR, the closer to 0 it will be. WFC is not a measure we can easily optimize. We would either have to enumerate the whole language model or use external non-ASR knowledge to predict continuations of the word sequence before the word in question has started. This would increase EO. In principle, we are rather interested in accepting an increase in WFC, when we delay messages in order to decrease EO.

WFF however, can reach values below 0. It converges towards the negative average of word length as an incremental ASR improves. For non-incremental ASR it would be positive: the average distance beween the sentence end and word end. WFF is a measure we can strive to reduce by sending fewer (especially fewer wrong) messages.

Another property we might be interested in optimizing is the time it takes from the first correct hypothesis to stabilize to a final hypothesis. We compute this *correction time* as the difference in time between WFF and WFC.[5] A correction time of 0 indicates that there was no correction, i.e. the ASR was immediately correct about a word, something which we would like to happen as often as possible.

Note that these are measures for each word in each processed utterance, and we will use distributional parameters of these timing measures (means and standard deviations) as metrics for the performance of the incremental setups described later.

2.4 Summary of Measures

In this section, we first described measures that evaluate the overall *correctness* of incrementally produced ASR hypotheses, not taking into account their sequential nature. We then turned to the dynamics of how the current hypothesis evolves in a way which we consider important for a consumer of incremental ASR, namely the *overhead* that results from edits to the hypothesis. Finally, we looked at the timing of individual messages with regard to first correct (potentially unstable) occurrence (WFC) and stability (WFF). In the next section, we use the measures defined here to characterize the incremental performance of our ASR, before we discuss ways to improve incremental performance in section 4.

3 Setup, Corpora and Base Measurements

We use the large-vocabulary continuous-speech recognition framework Sphinx-4 (Walker et al., 2004) for our experiments, using the built-in LexTree decoder, extended by us to provide incremental results. We built acoustic models for German, based on a small corpus of spontaneous instructions in a puzzle building domain,[6] and the Kiel corpus of read speech (IPDS, 1994). We use a trigram language model that is based on the puzzle domain transcriptions. As test data we use 85 recordings of two speakers (unknown to the acoustic model) that speak sentences similar to those in the puzzle domain.

We do not yet use recognition rescoring to optimize for word error rate, but just the ASR's best hypotheses which optimize for low sentence error. Incremental rescoring mechanisms such as that of

[5] In figure 1, the correction time for "zwei" is $9 - 7 = 2$.

[6] Available from `http://www.voxforge.org/home/downloads/speech/`

SER (non-incremental)	68.2 %
WER (non-incremental)	18.8 %
r-correct (cropped)	30.9 %
p-correct (cropped)	53.1 %
edit overhead	90.5 %
mean word duration	0.378 s
WFC: mean, stddev, median	0.276 s, 0.186 s, 0.230 s
WFF: mean, stddev, median	0.004 s, 0.268 s, –0.06 s
immediately correct	58.6 %

Table 1: Base measurements on our data

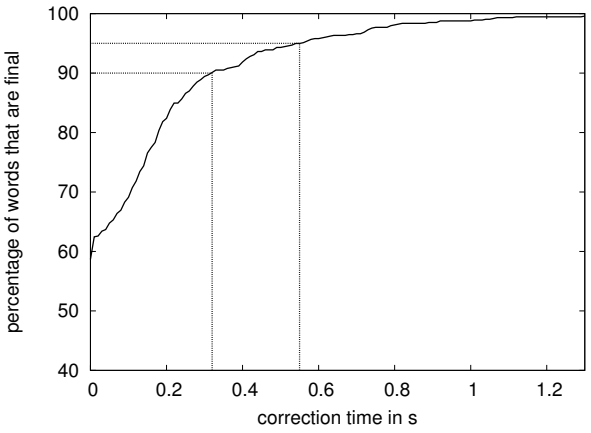


Figure 2: Distribution of correction times (WFF − WFC).

Razik et al. (2008) to optimize ASR performance are orthogonal to the approaches presented in section 4 and could well be incorporated to further improve incremental performance.

The individual recordings in our corpus are fairly short (5.5 seconds on average) and include a bit of silence at the beginning and end. Obviously, recognizing silence is much easier than recognizing words. To make our results more meaningful for continuous speech, we crop away all ASR hypotheses from before and after the active recognition process.[7] While this reduces our performance in terms of correctness (we crop away areas with nearly 100 % correctness), it has no impact on the edit overhead, as the number of changes in w_{curr} remains unchanged, and also no impact on the timing measures as all word boundaries remain the same.

3.1 Base Measurements

Table 1 characterises our ASR module (on our data) in terms of the metrics defined in section 2. Additionally we state *sentence error rate*, as the rate of sentences that contain at least one error, and word error rate computed in the usual way, as well as the mean duration of words in our corpus (as non-incrementally measured for our ASR).

We see that correctness is quite low. This is mostly due to the jitter that the evolving current hypothesis shows in its last few frames, jumping back and forth between highly-ranked alternatives. Also, our ASR only predicts words once there is acoustic evidence for several phonemes and every phoneme (being modelled by 3 HMM states) must have a duration of at least 3 frames. Thus, some errors relative to the final hypothesis occur because the ASR only hypothesizes about words once they already have a certain duration (and hence preceding hypotheses are not r-correct). The difference between r-correctness and p-correctness (20 % in our case) may be largely attributed to this fact.

The edit overhead of 90.5 % means that for every neccessary add message, there are nine superfluous (add or revoke) messages. Thus, a consumer of the ASR output would have to recompute its results ten times on average. In an incremental system, this consumer might itself output messages and further revise decisions as information from other modules becomes available, leading to a tremendous amount of changes in the system state. As ASR is the first module in an incremental spoken dialogue system, reducing the edit overhead is essential for overall system performance.

On average, the correct hypothesis about a word becomes available 276 ms after the word has started (WFC). With a mean word duration of 378 ms this means that information becomes available after roughly $\frac{3}{4}$ of the word have been spoken. Notice though that the median is somewhat lower than the mean, implying that this time is lower for most words and much higher for some words. In fact, the maximum for WFC in our data is 1.38 s.

On average, a word becomes final (i.e. is not changed anymore) when it has ended (mean(WFF) = 0.004). Again, the median is lower, indicating the unnormal distribution of WFF (more often lower, sometimes much higher).

Of all words, 58.6 % were immediately correctly

[7] In figure 1, hypotheses 1, 2 and 3 would be cropped away.

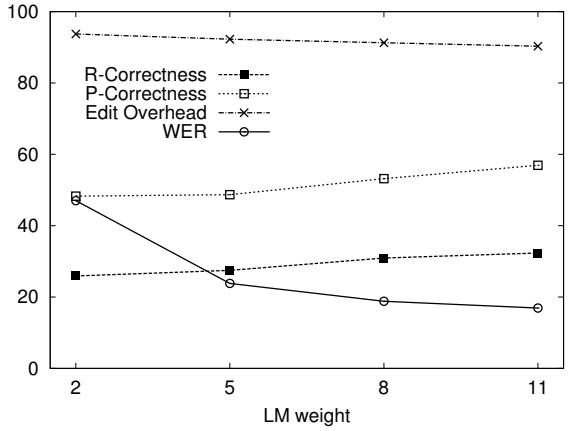


Figure 3: Correctness, Edit Overhead and Word Error Rate (WER) with varied language model weight and unaltered audio.

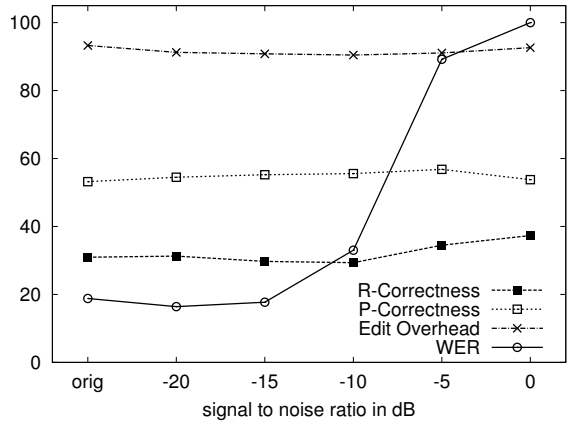


Figure 4: Correctness, Edit Overhead and Word Error Rate (WER) with additive noise (LM weight set to 8).

hypothesized by the ASR. Figure 2 plots the percentage of words with correction times equal to or lower than the time on the x-axis. While this starts at the initial 58.6 % of words that were immediately correct, it rises above 90 % for a correction time of 320 ms and above 95 % for 550 ms. Inversely this means that we can be certain to 90 % (or 95 %) that a current correct hypothesis about a word will not change anymore once it has not been revoked for 320 ms (or 550 ms respectively).

Knowing (or assuming with some certainty) that a hypothesis is final allows us, to *commit* ourselves to this hypothesis. This allows for reduced computational overhead (as alternative hypotheses can be abandoned) and is crucial if action is to be taken that cannot be revoked later on (as for example, initiating a response from the dialogue system). Figure 2 allows us to choose an *operating point* for commitment with respect to hypothesis age and certainty.

3.2 Variations of the Setup

In setting up our system we did not yet strive for best (non-incremental) performance; this would have required much more training material and parameter tweaking. We were more interested here in exploring general questions related to incremental ASR, and in developing approaches to improve incremental performance (see section 4), which we see as a problem that is independent from that of improving performance measures like (overall) accuracy.

To test how independent our measures are on details of the specific setting, such as quality of the audio material and of the language model, we varied these factors systematically, by adding white noise to the audio and changing the language model weight relative to the acoustic model. We varied the noise to produce signal to noise ratios ranging from hardly audible (−20 dB), through annoying noise (−10 dB) to barely understandable audio (0 dB).

Figure 3 gives an overview of the ASR-performance with different LM weights and figure 4 with degraded audio signals. Overall, we see that r-correctness and EO change little with different LM and AM performance and correspondigly degraded WER. A tendency can be seen that larger LM weights result in higher correctness and lower EO. A larger LM weight leads to less influence of acoustic events which dynamically change hypotheses, while the static knowledge from the LM becomes more important. Surprisingly, WER improved with the addition of slight noise, which we assume is due to differences in recording conditions between our test data and the training data of the acoustic model.

In the following experiments as well as in the data in table 1 above, we use a language model weight of 8 and unaltered audio.

4 Improving Incremental Performance

In the previous section we have shown how a standard ASR that incrementally outputs partial hypotheses after each frame processed performs with regard to our measures and showed that they remain

stable in different acoustic conditions and with differing LM weights. We now discuss ways of incrementally post-processing ASR hypotheses in order to improve selected measures.

We particularly look for ways to improve EO; that is, we want to reduce the amount of wrong hypotheses and resulting spurious edits that deteriorate later modules' performance, while still being as quick as possible with passing on relevant hypotheses. We are less concerned with correctness measures, as they do not capture well the dynamic evolution, which is important for further processing of the incremental hypothesis. We also discuss tradeoffs that are involved in the optimization decisions.

4.1 Right Context

Allowing the use of some *right context* is a common strategy to cope with incremental data. For example, our ASR already uses this strategy (with very short right contexts) internally at word boundaries to restrict the language model hypotheses to an acoustically plausible subset (Ortmanns and Ney, 2000). In the experiment described here, we allow the ASR a larger right context of size Δ by taking into account at time t the output of the ASR up to time $t - \Delta$ only. That is, what the ASR hypothesizes about the interval $]t - \Delta, t]$ is considered to be too immature and is discarded, and the hypotheses about the input up to $t - \Delta$ have the benefit of a lookahead up to t. This reduces jitter, which is found mostly to the very right of the incremental hypotheses. Thus, we expect to reduce the edit overhead in proportion with Δ. On the other hand, allowing the use of a right context leads to the current hypothesis *lagging behind* the gold standard. Correspondingly, WFC increases by Δ. Obviously, using only information up to $t - \Delta$ has averse effects on correctness as well, as this measure evaluates the word sequences up to w_{gold_t} which may already contain more words (those recognised in $]t - \Delta, t]$). Thus, to be more fair and to account for the lag when measuring the module's correctness, we additionally define *fair* r-correctness which restricts the evaluation up to time $t - \Delta$: $w_{hyp_t\, t-\Delta} = w_{gold_{t-\Delta}}$.

Figure 5 details the results for our data with right context between 1.5 s and -0.2 s. (The x-axis plots Δ as negative values, with 0 being "now". Results for a right context (Δ) of 1.2 can thus be found 1.2 to the left of 0, at -1.2.) We see that at least in the fair measure, fixed lag performs quite well at improving both the module's correctness and EO. This is due to the fact that ASR hypotheses become more and more stable when given more right context. Still, even for fairly long lags, many late edits still occur.

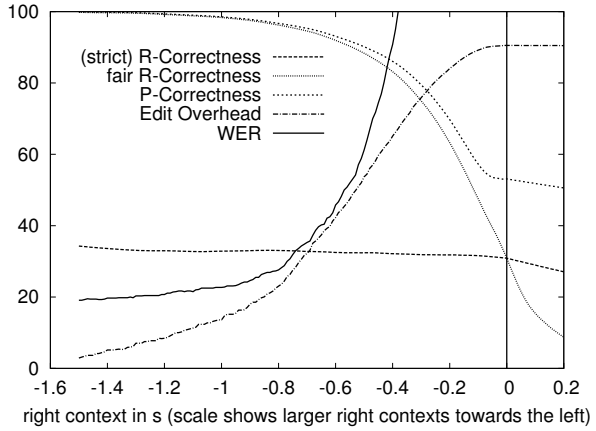


Figure 5: Correctness (see text), Edit Overhead and fixed-WER for varying right contexts Δ.

To illustrate the effects of a system that does not support edits of hypotheses, but instead commits right away, we plot WER that would be reached by a system that always commits after a right context of Δ. As can be seen in the figure, the WER remains higher than the non-incremental WER (18.8 %) even for fairly large right contexts. Also, the WER plot by Wachsmuth et al. (1998) looks very similar to ours and likewise shows a sweet spot suitable as an operating point with a right context of about 800 ms.

As expected, the analysis of timing measures shows an increase with larger right contexts with their mean values quickly approaching Δ (or $\Delta -$ mean word duration for WFF), which are the lower bounds when using right context. Correspondingly, the percentage of immediately correct hypotheses increases with right context reaching 90 % for $\Delta = 580$ ms and 98 % for $\Delta = 1060$ ms.

Finally, we can extend the concept of right context into negative values, predicting the future, as it were. By choosing a *negative* right context, in which we extrapolate the last hypothesis state by Δ into the future, we can measure the correctness of our hypotheses correctly predicting the close future, which is always the case when the current word is still be-

ing spoken. The graph shows that 15 % of our hypotheses will still be correct 100 ms in the future and 10 % will still be correct for 170 ms. Unfortunately, there is little way to tell apart hypotheses that will survive and those which will soon be revised.

4.2 Message Smoothing

In the previous section we reduced wrong edit messages by avoiding most of the recognition jitter by allowing the ASR a right context of size Δ, which directly hurt timing measures by roughly the same amount. In this section, we look at the sequence of partial hypotheses from the incremental ASR, using the dynamic properties as cues. We accomplish this by looking at the edit messages relative to the currently output word sequence. But instead of sending them to a consumer directly (updating the external word sequence), we require that an edit message be the result of N consecutive hypotheses. To illustrate the process with $N = 2$ we return to figure 1. None of the words "an", "ein" or "zwar" would ever be output, because they are only present for one time-interval each. Edit messages would be sent at the following times: $\oplus(\text{eins})$ at t_7, $\oplus(\text{zwei})$ at t_{10} (only then is "zwei" the result of two consecutive hypotheses) and $\oplus(\text{drei})$ at t_{13}. While no words are revoked in the example, this still occurs when a revocation is consecutively hypothesized for N frames.

We get controversial results for this strategy, as can be seen in figure 6: The edit overhead falls rapidly, reaching 50 % (for each message necessary, there is one superfluous message) with only 110 ms (and correspondingly increasing WFC by the same time) and 10 % with 320 ms. The same thresholds are reached through the use of right context at 530 ms and 1150 ms respectively as shown in figure 5. Likewise, the prefix correctness improvements are better than with using right context, but the r-correctness is poor, even under the "fair" measure. We believe this is due to correct hypotheses being held back too long due to the hypothesis sequence being interspersed with wrong hypotheses (which only last for few consecutive hypotheses) which reset the counter until the add message (for the prevalent and potentially correct word) is sent.[8]

[8]This could be resolved by using some kind of majority smoothing instead of requiring a message to be the result of *all* consecutive hypotheses. We will investigate this in future work.

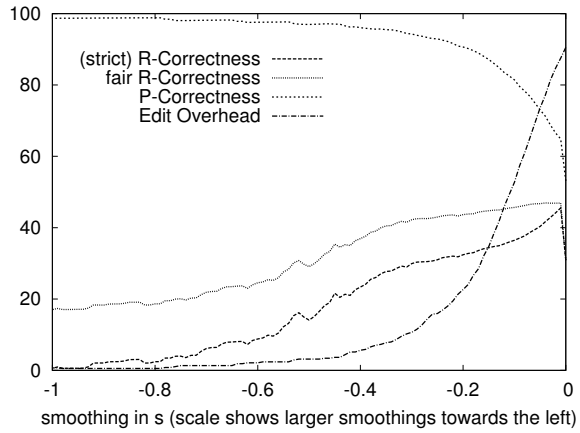


Figure 6: Correctness and Edit Overhead for varying smoothing lengths.

5 Conclusions and Further Directions

We have presented the problem of speech recognition for incremental systems, outlined requirements for incremental speech recognition and showed measures that capture how well an incremental ASR performs with regard to these measures. We discussed the measures and their implications in detail with our baseline system and showed that the incremental measures remain stable regardless of the specific ASR setting used.

Finally, we presented ways for the online post-processing of incremental results, looking for ways to improve some of the measures defined, while hurting the other measures as little as possible. Specifically, we were interested in generating less wrong hypotheses at the cost of possible short delays. While using right context shows improvements with larger delays, using message smoothing seems especially useful for fast processing. We think these two approaches could be combined to good effect. Together with more elaborate confidence handling a system could quickly generate hypotheses and then refine the associated confidences over time. We will explore this in future work.

Acknowledgments

This work was funded by a DFG grant in the Emmy Noether programme. We wish to thank the anonymous reviewers for helpful comments.

References

James Allen, George Ferguson, and Amanda Stent. 2001. An architecture for more realistic conversational systems. In *Proceedings of the Conference on Intelligent User Interfaces*, Santa Fe, USA.

IPDS. 1994. The Kiel Corpus of Read Speech. CD-ROM.

Anne Kilger and Wolfgang Finkler. 1995. Incremental generation for real-time applications. Technical Report RR-95-11, DFKI, Saarbrücken, Germany.

Stefan Ortmanns and Hermann Ney. 2000. Look-ahead techniques for fast beam search. *Computer Speech & Language*, 14:15–32.

Joseph Razik, Odile Mella, Dominique Fohr, and Jean-Paul Haton. 2008. Frame-Synchronous and Local Confidence Measures for on-the-fly Automatic Speech Recognition. In *Proceedings of Interspeech 2008*.

Sven Wachsmuth, Gernot A. Fink, and Gerhard Sagerer. 1998. Integration of parsing and incremental speech recognition. In *Proceedings of the European Signal Processing Conference*, volume 1, pages 371–375, Rhodes, Greece.

Willi Walker, Paul Lamere, Philip Kwok, Bhiksha Raj, Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel. 2004. Sphinx-4: A flexible open source framework for speech recognition. Technical Report SMLI TR2004-0811, Sun Microsystems Inc.

Steve Young, NH Russell, and JHS Thornton. 1989. Token passing: a simple conceptual model for connected speech recognition systems. *Cambridge University Engineering Department Technical Report CUED/F-INFENG/TR*, 38.

Geo-Centric Language Models for Local Business Voice Search

Amanda Stent, Ilija Zeljković, Diamantino Caseiro and **Jay Wilpon**
AT&T Labs – Research
180 Park Avenue Bldg. 103
Florham Park, NJ 07932, USA
stent, ilija, caseiro, jgw@research.att.com

Abstract

Voice search is increasingly popular, especially for local business directory assistance. However, speech recognition accuracy on business listing names is still low, leading to user frustration. In this paper, we present a new algorithm for geo-centric language model generation for local business voice search for mobile users. Our algorithm has several advantages: it provides a language model for any user in any location; the geographic area covered by the language model is adapted to the local business density, giving high recognition accuracy; and the language models can be pre-compiled, giving fast recognition time. In an experiment using spoken business listing name queries from a business directory assistance service, we achieve a 16.8% absolute improvement in recognition accuracy and a 3-fold speedup in recognition time with geo-centric language models when compared with a nationwide language model.

1 Introduction

Voice search is an increasingly popular application of speech recognition to telephony. In particular, in the last two years several companies have come out with systems for local business voice search (LBVS). In this type of application, the user provides a desired location (city/state) and a business name, and the system returns one or more matching business listings. The most traditional LBVS applications are commercial 411 services, which are implemented as a speech-only two-exchange dialog such as the one in Figure 1. In this approach to LBVS, the speech recognizer (ASR) uses one grammar to recognize city/state, and then uses separate grammars for recognizing listings in each local area. This gives relatively high recognition accuracy.

Advancements in ASR and search technology have made a more information retrieval-style LBVS feasible. In this approach, the ASR typically uses a large stochastic language model that permits the user to specify location and listing name or category together in a single utterance, and then submits recognition results to a search engine (Natarajan et al., 2002). This gives the user more flexibility to "say anything at any time". However, in recent evaluations of one-exchange LBVS we have found that locations are recognized with much higher accuracy than listing names[1]. This may mean that the user has to repeat both location and listing several times (while in a traditional two-exchange interaction only one piece of information would have to be repeated). In effect, system developers have traded recognition accuracy for interaction flexibility, potentially increasing user frustration.

Advances in mobile phone technology make it possible for us to combine the advantages of two-exchange and one-exchange LBVS. The newest smart phones come with global positioning system (GPS) receivers and/or with the ability to determine location through cell tower triangulation or wi-fi. If we know the location of a LBVS user, we can use a **geo-centric language model** to achieve improved speech recognition accuracy and speed. This approach unobtrusively exploits the benefits of two-

[1]The vocabulary size for listing names is larger than that for cities and states in the USA.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 389–396,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

S City and state?
U Glendale California
S What listing?
U pizza

Figure 1: Example 411-search dialog

exchange voice search applications, while maintaining the flexibility of one-exchange systems.

In this paper, we present an efficient algorithm for constructing geo-centric language models from a business listing database and local business search logs. Our algorithm has several advantages: it provides a language model for any user in any location; the geographic area covered by the language model is adapted to the local business density, giving high recognition accuracy; and the language models can be pre-compiled, giving fast recognition time. In an experiment using LBVS queries, we achieve: a 16.8% absolute improvement in recognition accuracy and a 3-fold speedup in recognition time with geo-centric language models when compared with a nationwide language model (such as those used in one-exchange LBVS); and a 4.4% absolute increase in recognition accuracy and a 16% speedup in recognition time with geo-centric language models when compared with local area language models (such as those used in two-exchange LBVS).

The rest of this paper is structured as follows: In Section 2 we discuss related work on voice-driven local search. In Section 3 we present the motivation for and architecture of a LBVS application. In Section 4 we present our algorithm for generating geo-centric language models. In Section 5 we describe an evaluation of the performance of our geo-centric language models on business listing name queries from a deployed voice-driven search application. In Section 6 we conclude and present future work.

2 Related Work

LBVS is the most recent variation on automated directory assistance (Buntschuh et al., 1998). ASR for directory assistance is difficult for several reasons: the vocabulary is large and includes foreign words; there may be multiple possible pronunciations for many words; and the frequency distribution of words in the vocabulary is unusual, with a few words occurring very often and the rest, rarely. These difficulties are compounded by directory size. For example, Kamm *et al.* (1995), in experiments on personal name directories, showed that ASR accuracy decreases from 82% for a 200 name directory to 16.5% for a 1.5 million name directory.

One way to reduce the directory size is to cover a smaller geographic area. For example, early LBVS covered only one city (Seide and Kellner, 1997; Collingham et al., 1997). Later, two-exchange, applications required the user to specify their desired location in the first exchange. This information was then used to select a local area grammar or language model for recognition of the listing name (Acero et al., 2008; Bacchiani et al., 2008; Yu et al., 2007; Georgila et al., 2003). In our research, we have created a novel method for constructing language models that cover a very small geographic area specific to the user's geo-location.

Another way to reduce the directory size is to drop listings that are unlikely to be requested. For example, Kamm *et al.* (1995), in their analysis of 13,000 directory assistance calls, found that a mere 245 listings covered 10% of the call volume, and 870 listings covered 20%. Chang *et al.* (2008) found that in their data sets, 19-25% of the call volume was covered by the top 200 listings. We take a different approach: we add frequent nationwide listings to our geo-centric language models to increase coverage.

Other work related to ASR in automated directory assistance has looked at ways in which users refer to locations (Gupta et al., 1998) and listings (Li et al., 2008; Scharenborg et al., 2001; Yu et al., 2007), confidence scoring for directory assistance search results (Wang et al., 2007), and ways of handling recognition errors through multimodal confirmation and correction (Acero et al., 2008; Chang et al., 2008; Paek et al., 2008). We do not address these issues here.

3 Local Business Voice Search

The current generation of smart phones contains GPS and/or can run applications that can detect the user's geo-location using cell tower triangulation or wi-fi. We hypothesize that this geo-location information can be used in mobile LBVS to improve recognition accuracy without sacrificing interaction flexibility. Our analysis of a directory assistance data set shows that in the majority of cases, users

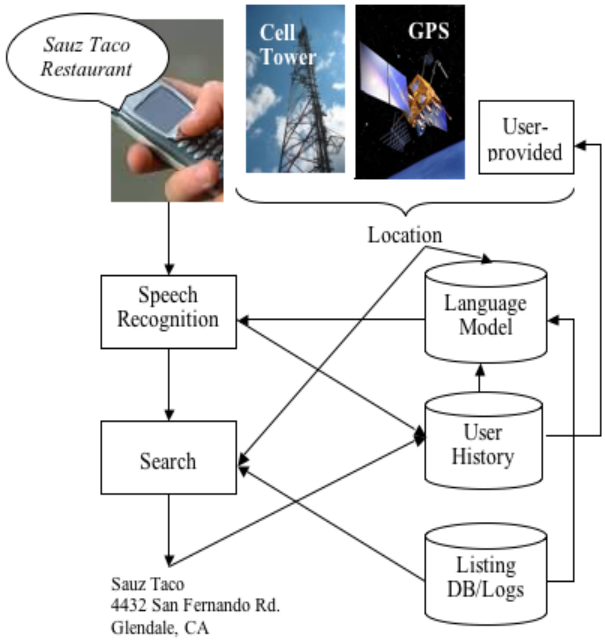


Figure 2: Architecture of a voice-driven local search application

request local listings. It is frustrating for the user of a LBVS who cannot retrieve information for a business right around the corner. So, a LBVS should maximize accuracy for local listings[2].

Figure 2 shows the architecture of a mobile LBVS. It includes ASR (in a speech-only or multimodal interface), search, and presentation of results (through speech, text and/or graphics). It also includes location information from GPS, cell tower triangulation or wi-fi, or the user's query history (from previous dialogs, or previous turns in this dialog).

4 Using Location to Tailor Language Models

There are two ways to use geo-location information in ASR for LBVS. One way is to use the user's geo-location to automatically determine the nearest city. City and state can then be used to select a **local area** language model (LM) for recognizing listing names. The advantages of this approach include: human knowledge about location can be included in the design of the local areas; and local areas can be designed to produce a minimal number of local area LMs. However, if the user is near the edge of the pre-defined local area, the selected LM may exclude businesses close to the user and include businesses far away from the user. Also, some local area LMs contain many more directory listings than others.

Another way is to construct a **geo-centric** LM covering businesses in a given radius around the user's geo-location. This approach has the advantage that listings included in the language model will certainly be close to the user. However, on-the-fly computation of geo-centric language models for large numbers of users is too computationally demanding given current database and processing technology. It is equally impractical to pre-compile all possible geo-centric language models, since commercial GPS provides coordinates accurate to about 20 feet. Here we present an algorithm for approximating true geo-centric language modeling in a way that is computationally feasible and user relevant.

4.1 Local Area Language Models

Telecommunications companies have long understood that customers may not know the exact town in which a desired listing is, or may be interested in listings from several nearby towns. Considerable effort has been devoted to defining local service areas (LSAs) for telephone directories. In the directory service that provided the database we use, business listings are organized into about 2000 LSAs, each consisting either of several adjacent small towns or of one big city. For example, the Morristown, NJ LSA includes Morristown itself as well as 53 adjacent localities and neighborhoods spanning from Pine Brook in the north-east to Mendham in the south-west. By contrast, the New York, NY LSA contains only New York City, which includes several hundred neighborhoods. The Morristown, NJ LSA contains 50000 business listings while the New York, NY LSA contains more than 200000 listings.

We construct one LM for each LSA, giving roughly 2000 local area LMs for the whole of the USA.

4.2 Geo-Centric Language Models

To construct a a geo-centric LM for a user, we need geo-coordinates (for the center of the LM) and a search radius (to determine the extent of the LM). It

[2]Of course, a LBVS should also give the user the option of specifying a different location, and/or should be able to recognize listings users are most likely to ask for that may not exist in their local area.

Figure 3: Geo-centric areas in New York City

Figure 4: Geo-centric area near Vaughn, Montana

is computationally infeasible to either pre-compute geo-centric LMs for each uniquely identifiable set of geo-coordinates in the USA, or to compute them on-the-fly for large numbers of users. Fortunately, the number of business geo-coordinates in the USA is much sparser than the number of possible user geo-coordinates. There are about 17 million name-address unique businesses in the USA; assuming 8-digit geo-code accuracy they are located at about 8.5 million unique geo-coordinates[3]. So we build LMs for business geo-coordinates rather than user geo-coordinates, and at run-time we map a user's geo-coordinates to those of their closest business.

To determine the search radius, we need a working definition of "local listing". However, "local" varies depending on one's location. In New York City, a local listing may be one up to ten blocks away (covering a smaller geographic area than the LSA), while in Montana a local listing may be one that one can drive to in 45 minutes (covering a larger geographic area than the LSA). Compare Figures 3 and 4. "Local" is clearly related to business density at a particular location. So we compute business density and use this to determine the radius of our geo-centric LMs.

We can do even better than this, however. Businesses are clustered geographically (in towns, shopping malls, etc.). This means that the set of listings local to one business is likely to be very similar to the set of listings local to a nearby business. So we do not need to build a separate LM for each business listing; instead, we can pre-determine the number of businesses we want to be different from one LM to another. Then we can "quantize" the business geo-coordinates so that those that have fewer than that number of businesses different between their search radii end up sharing a single LM.

Our algorithm for constructing geo-centric LMs starts with LSAs. It proceeds in two stages: first, the business centers for the LMs are found. Second, a search radius is computed for each LM center; and third, the data for the LM is extracted.

The LM center finding algorithm uses two parameters: r_1 (radius within an LSA; should be a little smaller than average LSA radius) and N_q (number of businesses that should be different between two different geo-centric LMs). For each LSA:

1. *Find mean latitude and longitude for the LSA:* Compute mean and standard deviation for latitude (μ_{lb}, σ_{lb}) and longitude (μ_{gb}, σ_{gb}) over all businesses in the LSA.

2. *Exclude national businesses which are listed in the LSA with their out-of-LSA address and geo-coordinates:* Compute mean and standard deviation of latitude and longitude, (μ_l, σ_l) and (μ_g, σ_g) respectively, using all geo-coordinates (l, g) where: (l, g) is within a r_1-mile radius of (μ_{lb}, μ_{gb}); l is within σ_{lb} of μ_{lb}; and g is within σ_{gb} of μ_{gb}.

3. *Compute business density in the most business-dense region in the LSA:* find a minimum and maximum longitude (g_m, g_M) and latitude (l_m, l_M) for all businesses that are within $(\pm\frac{1}{2}\sigma_g)$ and $(\pm\frac{1}{2}\sigma_l)$ of μ_g and μ_l respectively. Business density per square mile (d_2) is equal to the number of businesses in the rectangle defined by the low-left (g_m, l_m) and upper-right (g_M, l_M) corner. Business density per mile is $d_1 = \sqrt{d_2}$.

[3]The area of the USA with the highest business density is New York, NY, where about 270000 businesses share about 43000 geo-coordinates.

4. *Compute geo-location quantization accuracy:* Choose a desired number of business listings N_q that will fall to the same geo-coordinates when the quantization is applied. This corresponds roughly to the minimum desired number of different businesses in two adjacent geo-centric LMs. Quantization accuracy, in miles, δ_{qm}, then follows from the business density d_1: $\delta_{qm} = N_q/d_1$. Quantization accuracy for the longitude δ_g satisfies equation $distance((\mu_g, \mu_l), (\mu_g+\delta_g, \mu_l)) = \delta_{qm}$. δ_l satisfies a similar equation.

5. *Quantize geo-coordinates for each business in the LSA:* Compute quantized geo-coordinates (l_q, g_q) for each business in the LSA. $g_q = int(g/\delta_g)\times\delta_g$; $l_q = int(l/\delta_l)\times\delta_l$. Each unique (l_q, g_q) is a LM center.

The LM radius finding algorithm also uses two parameters: r_2 (maximum search radius for an LM); and N_p (minimum number of businesses within a geo-centric language model, should be smaller than average number of businesses per LSA). For each LM center:

1. Count the number of businesses at 1-mile radius increments of the LM center

2. Choose the smallest radius containing at least N_p listings (or the r_2 radius if there is no smaller radius containing at least N_p listings)

3. Extract data for all listings within the radius. Build LM from this data.

The number of geo-centric LMs can be arbitrarily small, depending on the parameter values. We believe that any number between 10K and 100K achieves good accuracy while maintaining tractability for LM building and selection. In the experiments reported here we used $r_1 = 3.5$, $N_q = 50$, $r_2 = 3$ and $N_p = 1000$, giving about 15000 LMs for the whole USA.

To summarize: we have described an algorithm for building geo-centric language models for voice-driven business search that: gives a local language model for any user anywhere in the country; uses business density determine "local" for any location in the country; can be pre-compiled; and can be tuned (by modifying the parameters) to maximize performance for a particular application

5 Experiments

In this section we report an evaluation of geo-centric language models on spoken business listing queries from an existing directory assistance application. We compare the recognition accuracy and recognition speed for geo-centric LMs to those of local area LMs, of a national LM, and of combined LMs.

5.1 Data

Our test data comes from an existing two-exchange directory assistance application. It comprises 60,000 voice queries, each consisting of a city and state in the first exchange, followed by a business listing name in the second exchange.

We wanted to test using queries for which we know there is a matching listing in the city/state provided by the caller. So we used only the 15000 queries for which there was a match in our nationwide business listing database[4]. We categorized each query as **nationwide** or **local** by looking up the listing name in our database. We considered any listing name that occurred five or more times to be nationwide; the remaining listings were considered to be local. This method fails to distinguish between national chains and different companies that happen to have the same name. (However, from a recognition point of view any listing name that occurs in multiple locations across the country is in fact nationwide, regardless of whether the businesses to which it refers are separate businesses.) It is also quite strict because we used string equality rather than looser name matching heuristics. Example national queries include *Wal-mart* and *Domino's Pizza*. Example local queries include *Sauz Taco* (Glendale, CA); *Dempsey's Restaurant* (Adrian, MI); and *Concord Farmers Club* (Saint Louis, MO). Some queries contain street names, e.g. *Conoco on South Division*; *uh Wal-Mart on five thirty five*; and *Chuy's Mesquite Broiler off of Rosedale*.

For each query in our data, we say that its local area LM is the local area LM that comes from its

[4] A query matched an entry in our database if there was a business listing in our database starting with the listing name portion of the query, in the city/state from the location portion of the query.

city and state, and that contains its listing name. Its geo-centric LM is defined similarly.

5.2 Language Model Construction

We constructed two baseline LMs. The first is a **National** LM. To take advantage of the non-uniform distribution of queries to listings (see Section 2), we also build a **Top 2000** LM containing only information about the top 2000 most frequently requested listing names nationwide[5] . We expected this LM to perform poorly on its own but potentially quite well in combination with local LMs.

For national, top 2000, local area and geo-centric LMs, we build trigram Katz backoff language models using AT&T's Watson language modeling toolkit (Riccardi et al., 1996). The models are built using the listing names and categories in our nationwide listing database. Listing names are converted to sentences containing the listing name, street address, neighborhood and city/state.

We predict that location-specific LMs will achieve high accuracy on local listings but will not be very robust to national listings. So we also experiment with *combination* LMs: local area combined with top 2000; geo-centric combined with top 2000; local area combined with national; and geo-centric combined with national. We use two combination stategies: *count merging* and *LM union*.

5.2.1 Count Merging

The *count merging* approach can be viewed as an instance of maximum a posteriori (MAP) adaptation. Let hw be a n-gram ending in word w and with a certain context h, and let $c_L(hw)$ and $C_T(hw)$ be its counts in the geo-centric/local area corpus L and top 2000 corpus T respectively. Then $p(w|h)$ is computed as:

$$p(w|h) = \frac{\lambda_L c_L(hw) + (1-\lambda_L)c_T(hw)}{\lambda_L c_L(h) + (1-\lambda_L)c_T(h)} \quad (1)$$

where λ_L is a constant that controls the contribution of each corpus to the combined model. We applied this combination strategy to local area/geo-centric and top 2000 only, not to local area/geo-centric and nationwide.

[5]We computed listing frequencies from query logs and used listings from the left-hand side of the frequency distribution curve before it flattens out; there were about 2000 of these.

5.2.2 LM Union

The *LM union* approach uses a union of language models at runtime. Let $W = w_0 w_1 \ldots w_{|W|}$ be a sentence, $p_L(W)$ be the probability of W in the geo-centric/local area corpus L, and $p_T(W)$ be the probability of W in the top 2000/national corpus T. Then $p(W)$ is computed as:

$$p(W) = max(\lambda_L p_L(W), (1-\lambda_L)p_T(W)) \quad (2)$$

λ_L is a constant that controls the contribution of each corpus to the combined model. We applied this combination strategy to local area/geo-centric and top 2000, and to local area/geo-centric and nationwide.

Given the small size of our test set relative to the large number of local LMs it is unfeasible to train λ_L on held-out data. Instead, we selected a value for λ_L such that the adjusted frequency of the top business in the top 2000 corpus becomes similar to the frequency of the top business in the local LM. We anticipate that if we did have data for training λ_L more weight would be given to the local area/geo-centric LM.

5.3 Experimental Method

In our experiments we use AT&T's Watson speech recognizer with a general-purpose acoustic model trained on telephone speech produced by American English speakers (Goffin et al., 2005). We ran all tests on a research server using standard settings for our speech recognizer for large vocabulary speech recognition. For each LM we report recognition accuracy (string accuracy and word accuracy) overall, on nationwide listings only, on local listings only, and on queries that contain street names only. We also report recognition time (as a fraction of real time speed).

5.4 Results

Results are given in Table 1. Comparing the baseline (National LM) to our geo-centric LMs, we see that we achieve a 16.8% absolute increase in overall sentence accuracy with a 3-fold speedup. Most of the improvement in sentence accuracy is due to better performance on local queries; however, we also achieve a 2.9% absolute increase in sentence accuracy on nationwide queries.

LM	Recognition accuracy: String/Word [%]				Real time speed
	Overall	Nationwide queries	Local queries	Queries with street name	
Nationwide language models					
National	51.3/58.0	59.9/60.8	40.3/54.1	17.9/47.3	1.05
Top 2000	23.2/31.6	40.6/43.3	9.5/25.8	1.3/18.3	0.44
Local language models					
Local area	63.7/69.7	60.8/63.2	69.5/77.2	22.4/53.4	0.42
Geo-centric	68.1/73.0	62.8/65.0	75.0/81.7	15.1/49.7	0.36
Combined language models, LM union					
Local area, national	58.9/64.5	61.4/62.3	57.9/67.1	21.8/50.6	0.84
Geo-centric, national	64.7/69.1	63.6/64.5	67.2/74.5	23.2/52.1	0.78
Local area, top 2000	60.0/67.0	62.1/65.8	61.8/71.3	20.6/50.3	0.45
Geo-centric, top 2000	64.7/70.7	63.4/66.7	68.8/76.5	14.7/48.2	0.42
Combined language models, count merging					
Local area, top 2000	66.7/72.2	69.2/71.5	67.8/75.7	22.5/54.0	0.50
Geo-centric, top 2000	67.7/72.6	68.3/70.5	70.4/77.7	13.2/46.9	0.44

Table 1: Results on mobile 411 data (total listings 14235; national listings 4679; local listings 2495; listings with street addresses 1163)

Now we look at the performance of different approaches to nationwide and local language modeling. First we compare the two nationwide LMs. As expected, we see that the overall sentence accuracy for the National LM is more than twice as high as that of the Top 2000 LM, but the recognition time is more than twice as slow. Next we compare the two local language modeling approaches. We see that geo-centric LMs achieve a 4.4% absolute increase in overall sentence accuracy compared to local area LMs and a 5.5% increase in sentence accuracy on local listings, while using less processing time.

Next we look at combination language models. When we combine local and nationwide LMs using LM union, we get small increases in sentence accuracy for nationwide queries compared to local LMs alone. However, sentence accuracy for local listings decreases. Also, these models use more processing time than the local LMs. When we combine local and national LMs using count merging, we get larger increases in sentence accuracy for nationwide queries over local LMs alone, and smaller decreases for local queries, compared to using LM union. LMs trained using count merging use more processing time than those trained using LM union, but still less than the National LM.

We conclude that: geo-centric language modeling leads to increased recognition accuracy and improvements in recognition time, compared to using a national language model; geo-centric language modeling leads to increased recognition accuracy and improvements in recognition time, compared to using local area language models; and geo-centric language models can be combined with a "most frequently asked-for" nationwide language model to get increased recognition accuracy on nationwide queries, at the cost of a small increase in recognition time and a slight decrease in recognition accuracy for local listings.

Further analysis of our results showed another interesting phenomenon. While geo-centric LMs achieve higher recognition accuracy than the National LM and local area LMs on nationwide and local queries, recognition accuracy on queries that contain a street name decreases. The likely reason is that small local LMs do not have rich street name coverage and people often do not refer to a street address precisely. A person might use a route number instead of a street name; if a single road has different names at different points they might use the wrong name; or they might use a variation on the actual name. For example, the query "Conoco on South Divison" is correctly recognized by our national LM but not with a geo-centric LM. The closest matching listing in our database for that location is "Conoco Convenience Store on South Boulevard". We note that we did not make any attempt to generalize over the street names in our LMs, sim-

ply pulling one street name for each listing from the database. Slightly more robust handling of street names may cause this phenomenon to disappear.

6 Conclusions and Future Work

Smart phones are able to give system developers increasingly detailed information about their users. This information can and should be exploited to give improved robustness and performance in customer services. In this paper, we explored the use of location information (from GPS or cell tower triangulation) to improve ASR accuracy in LBVS. We presented an algorithm for geo-centric language model generation that: adapts to the local business density; enables good local listing coverage; and requires only a limited number of language models. We compared the performance of our geo-centric language modeling to an alternative "local" language modeling approach and to a nationwide language modeling approach, and showed that we achieve significant improvements in recognition accuracy (a 4.4% absolute increase in sentence accuracy compared to local area language modeling, and a 16.8% absolute increase compared to the use of a national language model) with significant speedup.

We are currently testing our geo-centric language models in a LBVS prototype. In future work, we will optimize the parameters in our algorithm for geo-centric LM computation and merging. We also plan to explore the impact of integrating language modeling with search, and to examine the impact of these different language modeling approaches on performance of a trainable dialog manager that takes n-best output from the speech recognizer.

References

A. Acero, N. Bernstein, R. Chambers, Y. C. Ju, X. Li, J. Odell, P. Nguyen, O. Scholz, and G. Zweig. 2008. Live search for mobile: web services by voice on the cellphone. In *Proceedings of ICASSP*, pages 5256–5259.

M. Bacchiani, F. Beaufays, J. Schalkwyk, M. Schuster, and B. Strope. 2008. Deploying GOOG-411: Early lessons in data, measurement, and testing. In *Proceedings of ICASSP*, pages 5260–5263.

B. Buntschuh, C. Kamm, G. Di Fabbrizio, A. Abella, M. Mohri, S. Narayanan, I. Zeljkovic, R. Sharp, J. Wright, S. Marcus, J. Shaffer, R. Duncan, and J. Wilpon. 1998. VPQ: a spoken language interface to large scale directory information. In *Proceedings of ICSLP*.

S. Chang, S. Boyce, K. Hayati, I. Alphonso, and B. Buntschuh. 2008. Modalities and demographics in voice search: Learnings from three case studies. In *Proceedings of ICASSP*, pages 5252–5255.

R. Collingham, K. Johnson, D. Nettleton, G. Dempster, and R. Garigliano. 1997. The Durham telephone enquiry system. *International Journal of Speech Technology*, 2(2):113–119.

K. Georgila, K. Sgarbas, A. Tsopanoglou, N. Fakotakis, and G. Kokkinakis. 2003. A speech-based human-computer interaction system for automating directory assistance services. *International Journal of Speech Technology*, 6:145–159.

V. Goffin, C. Allauzen, E. Bocchieri, D. Hakkani-Tur, A. Ljolje, S. Parthasarathy, M. Rahim, G. Riccardi, and M. Saraclar. 2005. The AT&T Watson speech recognizer. In *Proceedings ICASSP*.

V. Gupta, S. Robillard, and C. Pelletier. 1998. Automation of locality recognition in ADAS plus. In *Proceedings of IVITA*, pages 1–4.

C. Kamm, C. Shamieh, and S. Singhal. 1995. Speech recognition issues for directory assistance applications. *Speech Communication*, 17(3–4):303–311.

X. Li, Y. C. Ju, G. Zweig, and A. Acero. 2008. Language modeling for voice search: a machine translation approach. In *Proceedings of ICASSP*, pages 4913–4916.

P. Natarajan, R. Prasad, R. Schwartz, and J. Makhoul. 2002. A scalable architecture for directory assistance automation. In *Proceedings of ICASSP*, pages 21–24.

T. Paek, B. Thiesson, Y. C. Ju, and B. Lee. 2008. Search Vox: Leveraging multimodal refinement and partial knowledge for mobile voice search. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 141–150.

G. Riccardi, R. Pieraccini, and E. Bocchieri. 1996. Stochastic automata for language modeling. *Computer Speech and Language*, 10(4):265–293.

O. Scharenborg, J. Sturm, and L. Boves. 2001. Business listings in automatic directory assistance. In *Proceedings of Eurospeech*, pages 2381–2384.

F. Seide and A. Kellner. 1997. Towards an automated directory information system. In *Proceedings of Eurospeech*, pages 1327–1330.

Y. Y. Wang, D. Yu, Y. C. Ju, G. Zweig, and A. Acero. 2007. Confidence measures for voice search applications. In *Proceedings of INTERSPEECH*.

D. Yu, Y. C. Ju, Y. Y. Wang, G. Zweig, and A. Acero. 2007. Automated directory assistance system – from theory to practice. In *Proceedings of INTERSPEECH*, pages 2709–2712.

Improving the Arabic Pronunciation Dictionary for Phone and Word Recognition with Linguistically-Based Pronunciation Rules

Fadi Biadsy∗ and **Nizar Habash**† and **Julia Hirschberg**∗
∗Department of Computer Science, Columbia University, New York, USA
{fadi,julia}@cs.columbia.edu
†Center for Computational Learning Systems, Columbia University, New York, USA
habash@ccls.columbia.edu

Abstract

In this paper, we show that linguistically motivated pronunciation rules can improve phone and word recognition results for Modern Standard Arabic (MSA). Using these rules and the MADA morphological analysis and disambiguation tool, multiple pronunciations per word are automatically generated to build two pronunciation dictionaries; one for training and another for decoding. We demonstrate that the use of these rules can significantly improve both MSA phone recognition and MSA word recognition accuracies over a baseline system using pronunciation rules typically employed in previous work on MSA Automatic Speech Recognition (ASR). We obtain a significant improvement in absolute accuracy in phone recognition of 3.77%–7.29% and a significant improvement of 4.1% in absolute accuracy in ASR.

1 Introduction

The correspondence between orthography and pronunciation in Modern Standard Arabic (MSA) falls somewhere between that of languages such as Spanish and Finnish, which have an almost one-to-one mapping between letters and sounds, and languages such as English and French, which exhibit a more complex letter-to-sound mapping (El-Imam, 2004). The more complex this mapping is, the more difficult the language is for Automatic Speech Recognition (ASR).

An essential component of an ASR system is its pronunciation dictionary (lexicon), which maps the orthographic representation of words to their phonetic or phonemic pronunciation variants. For languages with complex letter-to-sound mappings, such dictionaries are typically written by hand. However, for morphologically rich languages, such as MSA,[1] pronunciation dictionaries are difficult to create by hand, because of the large number of word forms, each of which has a large number of possible pronunciations. Fortunately, the relationship between orthography and pronunciation is relatively regular and well understood for MSA. Moreover, recent automatic techniques for morphological analysis and disambiguation (MADA) can also be useful in automating part of the dictionary creation process (Habash and Rambow, 2005; Habash and Rambow, 2007) Nonetheless, most documented Arabic ASR systems appear to handle only a subset of Arabic phonetic phenomena; very few use morphological disambiguation tools.

In Section 2, we briefly describe related work, including the baseline system we use. In Section 3, we outline the linguistic phenomena we believe are critical to improving MSA pronunciation dictionaries. In Section 4, we describe the pronunciation rules we have developed based upon these linguistic phenomena. In Section 5, we describe how these rules are used, together with MADA, to build our pronunciation dictionaries for training and decoding automatically. In Section 6, we present results of our evaluations of our phone- and word-recognition systems (XPR and XWR) on MSA comparing these systems to two baseline systems, BASEPR and BASEWR.

[1] MSA words have fourteen features: part-of-speech, person, number, gender, voice, aspect, determiner proclitic, conjunctive proclitic, particle proclitic, pronominal enclitic, nominal case, nunation, idafa (possessed), and mood. MSA features are realized using both concatenative (affixes and stems) and templatic (root and patterns) morphology with a variety of morphological and phonological adjustments that appear in word orthography and interact with orthographic variations.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 397–405,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

We conclude in Section 7 and identify directions for future research.

2 Related Work

Most recent work on ASR for MSA uses a single pronunciation dictionary constructed by mapping every undiacritized word in the training corpus to all of the diacritized Buckwalter analyses and the diacritized versions of this word in the Arabic Treebank (Maamouri et al., 2003; Afify et al., 2005; Messaoudi et al., 2006; Soltau et al., 2007). In these papers, each diacritized word is converted to a single pronunciation with a one-to-one mapping using "very few" unspecified rules. None of these systems use morphological disambiguation to determine the most likely pronunciation of the word given its context. Vergyri et al. (2008) *do* use morphological information to predict word pronunciation. They select the top choice from the MADA (Morphological Analysis and Disambiguation for Arabic) system for each word to train their acoustic models. For the test lexicon they used the undiacritized orthography, as well as all diacritizations found for each word in the training data as possible pronunciation variants. We use this system as our baseline for comparison.

3 Arabic Orthography and Pronunciation

MSA is written in a morpho-phonemic orthographic representation using the *Arabic script*, an alphabet accented with optional diacritical marks.[2] MSA has 34 phonemes (28 consonants, 3 long vowels and 3 short vowels). The Arabic script has 36 basic letters (ignoring ligatures) and 9 diacritics. Most Arabic letters have a one-to-one mapping to an MSA phoneme; however, there are a small number of common exceptions (Habash et al., 2007; El-Imam, 2004) which we summarize next.

3.1 Optional Diacritics

Arabic script commonly uses nine optional diacritics: (a) three short-vowel diacritics representing the vowels /a/, /u/ and /i/; (b) one long-vowel diacritic (Dagger Alif ‘) representing the long vowel /A/ in a small number of words; (c) three *nunation* diacritics (F /an/, N /un/, K /in/) representing a combination of a short vowel and the nominal indefiniteness marker /n/ in MSA; (d) one consonant lengthening diacritic (called Shadda $\sim$) which repeats/elongates the previous consonant (e.g., $kat\sim ab$ is pronounced /kattab/); and (e) one diacritic for marking when there is no diacritic (called Sukun o).

Arabic diacritics can only appear *after* a letter. Word-initial diacritics (in practice, only short vowels) are handled by adding an extra Alif ا A (also called Hamzat-Wasl) at the beginning of the word. Sentence/utterance initial Hamzat-Wasl is pronounced like a glottal stop preceding the short vowel; however, the sentence medial Hamzat-Wasl is silent except for the short vowel. For example, *Ainkataba kitAbN* is /Ginkataba kitAbun/ but *kitAbN Ainkataba* is /kitAbun inkataba/. A 'real' Hamza (glottal stop) is always pronounced as a glottal stop. The Hamzat-Wasl appears most commonly as the Alif of the definite article *Al*. It also appears in specific words and word classes such as relative pronouns (e.g., *Aly* 'who' and verbs in pattern VII (Ain1a2a3).

Arabic short vowel diacritics are used together with the glide consonant letters w and y to denote the long vowels /U/ (as uw) and /I/ (iy). This makes these two letters ambiguous.

Diacritics are largely restricted to religious texts and Arabic language school textbooks. In other texts, fewer than 1.5% of words contain a diacritic. Some diacritics are lexical (where word meaning varies) and others are inflectional (where nominal case or verbal mood varies). Inflectional diacritics are typically word final. Since nominal case, verbal mood and nunation have all disappeared in spoken dialectal Arabic, Arabic speakers do not always produce these inflections correctly or at all.

Much work has been done on automatic Arabic diacritization (Vergyri and Kirchhoff, 2004; Ananthakrishnan et al., 2005; Zitouni et al., 2006; Habash and Rambow, 2007). In this paper, we use the MADA (Morphological Analysis and Disambiguation for Arabic) system to diacritize Arabic (Habash and Rambow, 2005; Habash and Rambow, 2007). MADA, which uses the Buckwalter Arabic morphological Analyzer databases (Buckwalter, 2004), provides the necessary information to determine Hamzat-Wasl through morphologically tagging the definite article; in most other cases it outputs the special symbol "{" for Hamzat-Wasl.

[2] We provide Arabic script orthographic transliteration in the Buckwalter transliteration scheme (Buckwalter, 2004). For Modern Standard Arabic phonological transcription, we use a variant of the Buckwalter transliteration with the following exceptions: glottal stops are represented as /G/ and long vowels as /A/, /U/ and /I/. All Arabic script diacritics are phonologically spelled out.

3.2 Hamza Spelling

The consonant Hamza (glottal stop /G/) has multiple forms in Arabic script: ء ′, أ >, إ <, ؤ &, ئ }, آ |. There are complex rules for Hamza spelling that primarily depend on its vocalic context. For example, ئ } is used word medially and finally when preceded or followed by an /i/ vowel. Similarly, the Hamza form آ | is used when the Hamza is followed by the long vowel /A/.

Hamza spelling is further complicated by the fact that Arabic writers often replace hamzated letters with the un-hamzated form (أ > → ا A) or use a two-letter spelling, e.g. ئ } → ىء Y'. Due to this variation, the un-hamzated forms (particularly for أ > and إ <) are ignored in Arabic ASR evaluation. The MADA system regularizes most of these spelling variations as part of its analysis.

3.3 Morpho-phonemic Spelling

Arabic script includes a small number of morphemic/lexical phenomena, some very common:

- **Ta-Marbuta** The Ta-Marbuta (p) is typically a feminine ending. It appears word-finally, optionally followed by a diacritic. In MSA it is pronounced as /t/ when followed by a diacritic; otherwise it is silent. For example, $maktabapN$ 'a library' is pronounced / maktabatun/.

- **Alif-Maqsura** The Alif-Maqsura (Y) is a silent derivational marker, which always follows a short vowel /a/ at the end of a word. For example, $rawaY$ 'to tell a story' is pronounced /rawa/.

- **Definite Article** The Arabic definite article is a proclitic that assimilates to the first consonant in the noun it modifies if this consonant is alveolar or dental (except for j). These are the so-called Sun Letters: *t, v, d, *, r, z, s, $, S, D, T, Z, l,* and *n*. For example, the word *Al$ams* 'the sun' is pronounced /a$$ams/ not */al$ams/. The definite article does not assimilate to the other consonants, the Moon Letters. For example, the word Alqamar 'the moon' is pronounced /alqamar/ not */aqqamar/.

- **Silent Letters** A silent Alif appears in the morpheme $+uwA$ /U/ which indicates masculine plural conjugation in verbs. Another silent Alif appears after some nunated nouns, e.g., kitaAbAF /kitAban/. In some poetic readings, this Alif can be produced as the long vowel /A/: /kitAbA/. Finally, a common odd spelling is that of the proper name $Eamrw$ /Eamr/ 'Amr'where the final w is silent.

4 Pronunciation Rules

As noted in Section 3, diacritization alone does not predict actual pronunciation in MSA. In this section we describe a set of rules based on MSA phonology which will extend a diacritized word to a set of possible pronunciations. It should be noted that even MSA-trained speakers, such as broadcast news anchors, may not follow the "proper" pronunciation according to Arabic syntax and phonology. So we attempt to accommodate these pronunciation variants in our pronunciation dictionary.

The following rules are applied on each diacritized word.[3] These rules are divided into four categories: (I) a shared set of rules used in all systems compared (BASEPR, BASEWR, XPR and XWR);[4] (II) a set of rules in BASEPR and BASEWR which we modified for XPR and XWR; (III) a first set of new rules devised for our systems XPR and XWR; and (IV) a second set of new rules that generate additional pronunciation variants. Below we indicate, for each rule, how many words in the training corpus (335,324 words) had their pronunciation affected by the rule.

I. Shared Pronunciation Rules

1. **Dagger Alif:** ‘ → /A/
   (e.g., h‘*A → hA*A) (This rule affected 1.8% of all the words in our training data)

2. **Madda:** | → /G A/
   (e.g., Al|n → AlGAn) (affected 1.9%)

3. **Nunation:** AF → /a n/, F → /a n/, /K/ → /i n/, N → /u n/
   (e.g., kutubAF → kutuban) (affected 9.7%)

4. **Hamza:** All Hamza forms: $', \}, \&, <, >$ → /G/
   (e.g., >kala → Gakala) (affected 21.3%)

[3] Our script that generates the pronunciation dictionaries from MADA output can be downloaded from *www.cs.columbia.edu/speech/software.cgi*.

[4] We have attempted to replicate the baseline pronunciation rules for (Vergyri et al., 2008) based on published work and personal communications with the authors.

5. **Ta-Marbuta:** p → /t/
   (e.g., madrasapa → madrasata) (affected 15.3%)

II. Modified Pronunciation Rules

1. **Alif-Maqsura:** Y → /a/
   (e.g., salomY → saloma) (affected 4.2%) *(Baseline: Y → /A/)*

2. **Shadda:** Shadda is always removed
   (e.g., ba$∼ara → ba$ara) (affected 23.8%) *(Baseline: the consonant was doubled)*

3. **U and I:** uwo → /U/, iyo → /I/
   (e.g., makotuwob → makotUb) (affected 25.07%) *(Baseline: same rule but it inaccurately interacted with the baseline Shadda rule)*

III. New Pronunciation Rules

1. **Waw Al-jamaa:** suffixes uwoA → /U/
   (e.g., katabuwoA → katabU) (affected 0.4%)

2. **Definite Article:** Al → /a l/ (if tagged as Al+ by MADA)
   (e.g., wAlkitAba → walkitAba) (affected 30.0%)

3. **Hamzat-Wasl:** { is always removed.
   (affected 3.0%)

4. **"Al" in relative pronouns:** Al → /a l/
   (affected 1.3%)

5. **Sun letters:** if the definite article (Al) is followed by a sun letter, remove the *l*.
   (e.g., Al$amsu → A$amsu) (affected 8.1%)

IV. New Pronunciation Rules Generating Additional Variants

- **Ta-Marbuta:** if a word ends with Ta-Marbuta (p) followed by any diacritic, remove the Ta-Marbuta and its diacritic. Apply the rules above (I-III) on the modified word and add the output pronunciation.
  (e.g., marbwTapF → marbwTa) (affected 15.3%)

- **Case ending:** if a word ends with a short vowel (a, u, i), remove the short vowel. Apply rules (I-III) on the modified word, and add the output pronunciation
  (e.g., yaktubu → yaktub (affected 60.9%)

As a post-processing step in all systems, we remove the Sukun diacritic and convert every letter X to phoneme /X/. In XPR and XWR, we also remove short vowels that precede or succeed a long vowel.

5 Building the Pronunciation Dictionaries

As noted above, pronunciation dictionaries map words to one or more phonetically expressed pronunciation variants. These dictionaries are used for training and decoding in ASR systems. Typically, most data available to train large vocabulary ASR systems is orthographically (not phonetically) transcribed. There are two well-known alternatives for training acoustic models in ASR: (1) bootstrap training, when some phonetically annotated data is available, and (2) flat-start, when such data is not available (Young et al., 2006). In flat-start training, for example, the pronunciation dictionary is used to map the orthographic transcription of the training data to a sequence of phonetic labels to train the initial monophone models. Next, the dictionary is employed again to produce networks of possible pronunciations which can be used in forced alignment to obtain the most likely phone sequence that matches the acoustic data. Finally, the monophone acoustic models are re-estimated. In our work, we refer to this dictionary as the **training pronunciation dictionary**. The second usage of the pronunciation dictionary is to generate the pronunciation models while decoding. We refer to this dictionary as the **decoding pronunciation dictionary**.

For languages like English, no distinction between decoding and training pronunciation dictionaries is necessary. However, as noted in Section 3, short vowels and other diacritic markers are typically not orthographically represented in MSA texts. Thus ASR systems typically do not output fully diacritized transcripts. Diacritization is generally not necessary to make the transcript readable by Arabic-literate readers. Therefore, entries in the decoding pronunciation dictionary consist of undiacritized words that are mapped to a set of phonetically-represented diacritizations. However, every entry in the training pronunciation dictionary is a fully diacritized word mapped to a set of possible context-dependent pronunciations. Particularly in the training step, contextual information for each word is available from the transcript, so, for our work, we can use the MADA morphological tagger to obtain the most likely diacritics. As a result, the speech signal is mapped to a more accurate representation

of the training transcript, which we hypothesize will lead to a better estimation of the acoustic models.

As noted in Section 1, pronunciation dictionaries for ASR systems are usually written by hand. However, Arabic's morphological richness makes it difficult to create a pronunciation dictionary by hand since there are a very large number of word forms, each of which has a large number of possible pronunciations. The relatively regular relationship between orthography and pronunciation and tools for morphological analysis and disambiguation such as MADA, however, make it possible to create such dictionaries automatically with some success.[5]

5.1 Training Pronunciation Dictionary

In this section, we describe an automatic approach to building a pronunciation dictionary for MSA that covers all words in the orthographic transcripts of the training data. First, for each word in each utterance, we run MADA to disambiguate the word based on its context in the transcript. MADA outputs all possible fully-diacritized morphological analyses, ranked by their likelihood, the MADA confidence score.[6] We thus obtain a fully-diacritized orthographic transcription for training. Second, we map the highest-ranked diacritization of each word to a set of pronunciations, which we obtain from the pronunciation rules described in Section 4. Since MADA may not always rank the best analysis as its top choice, we also run the pronunciation rules on the **second** best choice returned by MADA, when the difference between the top two choices is less than a threshold determined empirically (in our implementation we chose 0.2). In Figure 1, the training pronunciation dictionary maps the 2^{nd} column (the entry keys) to the 3^{rd} column.

We generate the baseline training pronunciation dictionary using only the baseline rules from Section 4. This dictionary also makes use of MADA, but it maps the MADA-diacritized word to only one pronunciation. The baseline training dictionary maps the 2^{nd} column (the entry keys) to **only** one pronunciation in the 3^{rd} column in Figure 1.

[5]The MADA system (Habash and Rambow, 2005; Habash and Rambow, 2007) reports 4.8% diacritic error rate (DER) on all diacritics and 2.2% (DER) when ignoring the last (inflectional) diacritic.

[6]In our training data, only about 1% of all words are not diacritized because of lack of coverage in the morphological analysis component.

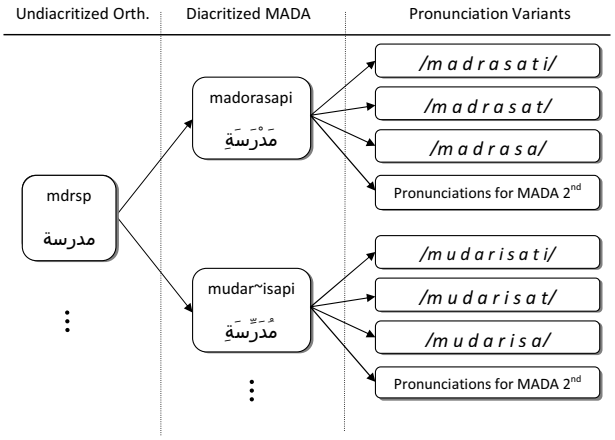


Figure 1: Mapping an undiacritized word to MADA outputs to possible pronunciations.

5.2 Decoding Pronunciation Dictionary

The decoding pronunciation dictionary is used in ASR to build the pronunciation models while decoding. Since, as noted above, it is standard to produce unvocalized transcripts when recognizing MSA, we must map word pronunciations to unvocalized orthographic output. Therefore, for each diacritized word in our training pronunciation dictionary, we remove diacritic markers and replace Hamzat-Wasl ({), <, and > by the letter 'A', and then map the modified word to the set of pronunciations for that word. For example, in Figure 1 the undiacritized word *mdrsp* in the 1^{st} column is mapped to the pronunciations in the 3^{rd} column. The baseline decoding pronunciation dictionary is constructed similarly from the baseline training pronunciation dictionary.

6 Evaluation

To determine whether our pronunciation rules are useful in speech processing applications, we evaluated their impact on two tasks, automatic phone recognition and ASR. For our experiments, we used the broadcast news TDT4 corpus (Arabic Set 1), divided into 47.61 hours of speech (89 news shows) for training and 5.18 hours (11 shows); test and training shows were selected at random. Both training and test data were segmented based on silence and non-speech segments and down-sampled to 8Khz.[7] This segmentation produced 20,707 speech segments for our training data and 2,255 segments for testing.

[7]One of our goals is phone recognition telephone conversation for Arabic dialect identifaction, hence the down-sampling.

6.1 Acoustic Models

Our monophone acoustic models are built using 3-state continuous HMMs without state-skipping with a mixture of 12 Gaussians per state. We extract standard MFCC (Mel Frequency Cepstral Coefficients) features from 25 ms frames, with a frame shift of 10 ms. Each feature vector is 39D: 13 features (12 cepstral features plus energy), 13 deltas, and 13 double-deltas. The features are normalized using cepstral mean normalization. For our ASR experiments, tied context-dependent cross-word triphone HMMs are created with the same settings as monophones. The acoustic models are speaker- and gender-independent, trained using ML (maximum likelihood) with flat-start.[8] We build our framework using the HMM Toolkit (HTK) (Young et al., 2006).

6.2 Phone Recognition Evaluation

We hypothesize that improved pronunciation rules will have a profound impact on phone recognition accuracy. To compare our phone recognition (XPR) system with the baseline (BASEPR), we train two phone recognizers using HTK. The BASEPR recognizer uses the training-pronunciation dictionary generated using the baseline rules; the XPR system uses a pronunciation dictionary generated using these rules plus our modified and new rules (cf. Section 5). The two systems are identical except for their pronunciation dictionaries.

We evaluate the two systems under two conditions: (1) phone recognition with a bigram phone language model (LM)[9] and (2) phone recognition with an open-loop phone recognizer, such that any phoneme can follow any other phoneme with a uniform distribution. Results of this evaluation are presented in Table 1.

Ideally, we would like to compare the performance of these systems against a common MSA phonetically-transcribed gold standard. Unfortunately, to our knowledge, such a data set does not exist. So we approximate such a gold standard on a blind test set through forced alignment, using the trained acoustic models and pronunciation dictionaries. Since our choice of acoustic model (of BASEPR or XPR) and pronunciation dictionary (again of BASEPR or XPR) can bias our results, we consider four *gold* variants (GV) with different combinations of acoustic model and pronunciation dictionary, to set expected lower and upper bounds. These combinations are represented in Table 1 as GV1–4, where the source of acoustic models is BASEPR or XPR and source of pronunciation rules are BASEPR, XPR or XPR and BASEPR combined. These GV are described in more detail below, as we describe our results.

Since BASEPR system uses a pronunciation dictionary with a one-to-one mapping of orthography to phones, the GV1 phone sequence for any test utterance's orthographical transcript according to BASEPR can be obtained directly from the orthographic transcript. Note that if, in fact, GV1 does represent the true gold standard (i.e., the correct phone sequence for the test utterances) then if XPR obtains a lower phone error rate using this gold standard than BASEPR does, we can conclude that in fact XPR's acoustic models are better estimated. This is in fact the case. In Table 1, first line, we see that XPR under both conditions (open-loop and bigram LM) significantly (p-value $< 2.2e-16$) outperforms the corresponding BASEPR phone recognizer using GV1.[10]

If GV1 does *not* accurately represent the phone sequences of the test data, then there must be some phones in the GV1 sequences that should be deleted, inserted, or substituted. On the hypothesis that our training-pronunciation dictionary might improve the BASEPR assignments, we enrich the baseline pronunciation dictionary with XPR's dictionary. Now, we force-align the orthographic transcript using this extended pronunciation dictionary, still using BASEPR's acoustic models, with the acoustic signal. We denote the output phone sequences as GV2. If a pronunciation generated using the BASEPR dictionary was already correct (in GV1) according to the acoustic signal, this forced alignment process still has the option of choosing it. We hypothesize that the result, GV2, is a more accurate representation of the true phone sequences in the test data, since it should be able to model the acoustic signal more accurately. On GV2, as on GV1, we see that XPR, under both conditions, significantly (p-

[8] Since our focus is a comparison of different approaches to pronunciation modeling on Arabic recognition tasks, we have not experimented with different features, parameters, and different machine learning approaches (such as discriminative training and/or the combination of both).

[9] The bigram phoneme LM of each phone recognizer is trained on the phonemes obtained from forced aligning the training transcript to the speech data using that recognizer's training pronunciation dictionary and acoustic models.

[10] Throughout this discussion we use paired t-tests to measure significant difference, where the sample values are the phone recognizer accuracies on the utterances.

Gold Variants			*Open-loop (Accuracy)*		*Bigram Phone LM (Accuracy)*	
GV	*Acoustic Model of*	*Pron. Dict. of*	BASEPR	XPR	BASEPR	XPR
1	BASEPR	BASEPR	37.40	39.21	41.56	45.17
2	BASEPR	BASEPR+XPR	38.64	42.41	43.44	50.73
3	XPR	XPR	37.06	42.38	42.21	51.41
4	XPR	BASEPR+XPR	37.47	42.74	42.59	51.51

Table 1: Comparing the effect of BASEPR and XPR pronunciation rules, alone and in combination, using 4 Gold Variants under two conditions (Open-loop and LM)

value $< 2.2e-16$) outperforms the corresponding BASEPR phone recognizers (see Table 1, second line).

We also compared the performance of the two systems using upper bound variants. For GV3 we used the forced alignment of the orthographic transcription using only XPR's pronuncation dictionary with XPR's acoustic models. In GV4 we combine the pronunciation dictionary of XPR with BASEPR dictionary and use XPR's acoustic models. Unsurprisingly, we find that the XPR recognizer significantly (p-value $<2.2e-16$) outperforms BASEPR when using these two variants under both conditions (see Table 1, third and fourth lines).

The results presented in Table 1 compare the robustness of the acoustic models as well as the pronunciation components of the two systems. We also want to evaluate the accuracy of our pronunciation predictions in representing the actual acoustic signal. One way to do this is to see how often the forced alignment process choose phone sequences using the BASEPR pronunciation dictionary as opposed to XPR's. We forced aligned the test transcript — using the XPR acoustic models and only the XPR pronunciation dictionary — with the acoustic signal. We then compare the output sequences to the output of the forced alignment process where the **combined** pronunciations from BASEPR+XPR and the XPR acoustic models were used. We find that the difference between the two is only 1.03% (with 246,376 phones, 557 deletions, 1696 substitutions, and 277 insertions). Thus, adding the BASEPR rules to XPR does not appear to contribute a great deal to the representation chosen by forced alignment. In a similar experiment, we use the BASEPR acoustic models instead of the XPR models and compare the results of using BASEPR-pronunciation dictionary with the combination of XPR+BASEPR's dictionaries for forced alignment. Interestingly, in this experiment we *do* find a significantly larger difference between the two outputs 17.04% (with 233,787 phones, 1404 deletions, 14013 substitutions, and 27040 insertions). We can hypothesize from these experiments that the baseline pronunciation dictionary alone is not sufficient to represent the acoustic signal accurately, since large numbers of phonemes are edited when adding the XPR pronunciations. In contrast, adding the BASEPR's pronunciation dictionary to XPR's shows a relatively small percentage of edits, which suggests that the XPR pronunciation dictionary extends and covers more accurately the pronunciations already contained in the BASEPR dictionary.

6.3 Speech Recognition Evaluation

We have also conducted an ASR experiment to evaluate the usefulness of our pronunciation rules for this application.[11] We employ the baseline pronunciation rules to generate the baseline training and decoding pronunciation dictionaries. Using these dictionaries, we build the baseline ASR system (BASEWR). Using our extended pronunciation rules, we generate our dictionaries and train our ASR system (XWR). Both systems have the same model settings, as described in Section 6.1. They also share the same language model (LM), a trigram LM trained on the undiacritized transcripts of the training data and a subset of Arabic gigawords (approximately 281 million words, in total), using the SRILM toolkit (Stolcke, 2002).

Table 2 presents the comparison of BASEWR with the XWR system. In Section 5.1, we noted that the top two choices from MADA may be included in the XWR pronunciation dictionary when the difference in MADA confidence scores for these two is less than a given threshold. So we analyze the impact of including this second MADA option in both the training and decoding dictionaries on ASR results. In all cases, whether the second MADA choice

[11] It should be noted that we have not attempted to build a state-of-the-art Arabic speech recognizer; our goal is purely to evaluate our approach to pronunciation modeling for Arabic.

is included or not, XWR significantly (p-values < 8.1e-15) outperforms BASEWR. Our best results are obtained when we include the top first and second MADA option in the decoding pronunciation dictionary but **only** the top MADA choice in the training pronunciation dictionary. The difference between this version of XWR and an XWR version which includes the top second MADA choice in the training dictionary is significant (p-value = 0.017).

To evaluate the impact of the set of rules that generate additional pronunciation variants (described in Section 4 - IV) on word recognition, we built a system, denoted as XWR_I-III, that uses only the first three sets of rules (I–III) and compared its performance to that of both BASEWR and the corresponding XWR system. As shown in Table 2, we observe that XWR_I-III significantly outperforms BASEWR in 2.27 (p-value < 2.2e-16). Also, the corresponding XWR that uses all the rules (including IV set) significantly outperforms XWR_I-III in 1.24 (p-value < 2.2e-16).

The undiacritized vocabulary size used in our experiment was 34,511. We observe that 6.38% of the words in the test data were out of vocabulary (OOV), which may partly explain our low absolute recognition accuracy. The dictionary size statistics (for entries generated from the training data only) used in these experiments are shown in Table 3. We have done some error analysis to understand the reason behind high absolute error rate for both systems. We observe that many of the test utterances are very noisy. We wanted to see whether XWR still outperforms BASEWR if we remove these utterances. Removing all utterances for which BASEWR obtains an accuracy of less than 25%, we are left with 1720/2255 utterances. On these remaining utterances, the BASEWR accuracy is 64.4% and XWR's accuracy is 67.23% — a significant difference despite the bias in favor of BASEWR.

System	Acc	Corr	Del	Sub	Ins
BASEWR	52.78	65.36	360	12297	4598
XWR_I–III (1 TD/DD)	55.05	66.84	324	11791	4308
XWR (1 TD/DD)	56.29	69.06	274	11031	4665
XWR (2 TD, 2 DD)	56.28	69.12	274	11008	4694
XWR (2 TD, 1 DD)	55.53	68.55	285	11206	4759
XWR (1 TD, 2 DD)	56.88	69.42	284	10891	4579

Table 2: Comparing the performance of BASEWR to XWR, where the top 1 or 2 MADA options are included in the training dictionary (TD) and decoding dictionary (DD). XWR I–III uses only the first three sets of pronunciation rules in Section 4. **Acc**uracy = (100 - WER); **Corr**ect is Accuracy without counting insertions (%). Total number of words is 36,538.

Dictionary	# entries	PPW
BASEPR TD	45,117	1
BASEPR DD	44,383	1.3
XPR TD (MADA top 1)	80,200	1.78
XPR TD (MADA top 1 and 2)	128,663	2.85
XWR DD (MADA top 1)	71,853	2.08
XWR DD (MADA top 1 and 2)	105,402	3.05

Table 3: Dictionary sizes generated fom the training data only (PPW: pronunciations per word, TD: Training pronunciation dictionary, DD: Decoding pronunciation dictionary).

7 Conclusion and Future Work

In this paper we have shown that the use of more linguistically motivated pronunciation rules can improve phone recognition and word recognition results for MSA. We have described some of the phonetic, phonological, and morphological features of MSA that are rarely modeled in ASR systems and have developed a set of pronunciation rules that encapsulate these features. We have demonstrated how the use of these rules can significantly improve both MSA phone recognition and MSA word recognition accuracy by a series of experiments comparing our XPR and XWR systems to the corresponding baseline systems BASEPR and BASEWR. We obtain an improvement in absolute accuracy in phone recognition of 3.77%–7.29% and a significant improvement of 4.1% in absolute accuracy in ASR.

In future work, we will address several issues which appear to hurt our recognition accuracy, such as handling the words that MADA fails to analyze. We also will develop a similar approach to handling dialectical Arabic speech using the MAGEAD morphological analyzer (Habash and Rambow, 2006). A larger goal is to employ the MSA and dialectical phone recognizers to aid in spoken Arabic dialect identification using phonotactic modeling (see (Biadsy et al., 2009)).

Acknowledgments

We are very thankful to Dimitra Vergyri for providing us with the details of the baseline pronunciation dictionary and for her useful suggestions. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023 (approved for public release, distribution unlimited). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

M. Afify, L. Nguyen, B. Xiang, S. Abdou, and J. Makhoul. 2005. Arabic broadcast news transcription using a one million word vocalized vocabulary. In *Proceedings of Interspeech 2005*, pages 1637–1640.

S. Ananthakrishnan, S. Narayanan, and S. Bangalore. 2005. Automatic diacritization of arabic transcripts for asr. In *Proceedings of ICON*, Kanpur, India.

F. Biadsy, J. Hirschberg, and N. Habash. 2009. Spoken Arabic Dialect Identification Using Phonotactic Modeling. In *Proceedings of EACL 2009 Workshop on Computational Approaches to Semitic Languages*, Athens, Greece.

T. Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Cat alog No.: LDC2004L02, ISBN 1-58563-324-0.

Y. A. El-Imam. 2004. Phonetization of Arabic: rules and algorithms. In *Computer Speech and Language 18*, pages 339–373.

N. Habash and O. Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580.

N. Habash and O. Rambow. 2006. MAGEAD: A Morphological Analyzer and Generator for the Arabic Dialects. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 681–688, Sydney, Australia.

N. Habash and O. Rambow. 2007. Arabic Diacritization through Full Morphological Tagging. In *Proceedings of the 8th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL07)*.

N. Habash, A. Soudi, and T. Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

M. Maamouri, A. Bies, H. Jin, and T. Buckwalter. 2003. Arabic treebank: Part 1 v 2.0. Distributed by the Linguistic Data Consortium. LDC Catalog No.: LDC2003T06.

A. Messaoudi, J. L. Gauvain, and L. Lamel. 2006. Arabic broadcast news transcription using a one million word vocalized vocabulary. In *Proceedings of ICASP 2006*, volume 1, pages 1093–1096.

H. Soltau, G. Saon, D. Povey, L. Mangu, B. Kingsbury, J. Kuo, M. Omar, and G. Zweig. 2007. The IBM 2006 GALE Arabic ASR System. In *Proceedings of ICASP 2007*.

S. Stolcke. 2002. Tokenization, Morphological Analysis, and Part-of-Speech Tagging for Arabic in One Fell Swoop. In *Proceedings of ICSLP 2002*, volume 2, pages 901–904.

D. Vergyri and K. Kirchhoff. 2004. Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition. In Ali Farghaly and Karine Megerdoomian, editors, *COLING 2004 Workshop on Computational Approaches to Arabic Script-based Languages*, pages 66–73, Geneva, Switzerland.

D. Vergyri, A. Mandal, W. Wang, A. Stolcke, J. Zheng, M. Graciarena, D. Rybach, C. Gollan, R. Schluter, K. Kirchhoff, A. Faria, and N. Morgan. 2008. Development of the SRI/Nightingale Arabic ASR system. In *Proceedings of Interspeech 2008*, pages 1437–1440.

S. Young, G. Evermann, M. Gales, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. 2006. The HTK Book, version 3.4: htk.eng.cam.ac.uk. Cambridge University Engineering Department.

I. Zitouni, J. S. Sorensen, and R. Sarikaya. 2006. Maximum Entropy Based Restoration of Arabic Diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 577–584, Sydney, Australia.

Using a maximum entropy model to build segmentation lattices for MT

Chris Dyer
Laboratory for Computational Linguistics and Information Processing
Department of Linguistics
University of Maryland
College Park, MD 20742, USA
redpony AT umd.edu

Abstract

Recent work has shown that translating segmentation lattices (lattices that encode alternative ways of breaking the input to an MT system into words), rather than text in any particular segmentation, improves translation quality of languages whose orthography does not mark morpheme boundaries. However, much of this work has relied on multiple segmenters that perform differently on the same input to generate sufficiently diverse source segmentation lattices. In this work, we describe a maximum entropy model of compound word splitting that relies on a few general features that can be used to generate segmentation lattices for most languages with productive compounding. Using a model optimized for German translation, we present results showing significant improvements in translation quality in German-English, Hungarian-English, and Turkish-English translation over state-of-the-art baselines.

1 Introduction

Compound words pose significant challenges to the lexicalized models that are currently common in statistical machine translation. This problem has been widely acknowledged, and the conventional solution, which has been shown to work well for many language pairs, is to segment compounds into their constituent morphemes using either morphological analyzers or empirical methods and then to translate from or to this segmented variant (Koehn et al., 2008; Dyer et al., 2008; Yang and Kirchhoff, 2006).

But into what units should a compound word be segmented? Taken as a stand-alone task, the goal of a compound splitter is to produce a segmentation for some input that matches the linguistic intuitions of a native speaker of the language. However, there are often advantages to using elements larger than single morphemes as the minimal lexical unit for MT, since they may correspond more closely to the units of translation. Unfortunately, determining the optimal segmentation is challenging, typically requiring extensive experimentation (Koehn and Knight, 2003; Habash and Sadat, 2006; Chang et al., 2008). Recent work has shown that by combining a variety of segmentations of the input into a *segmentation lattice* and effectively marginalizing over many different segmentations, translations superior to those resulting from any single single segmentation of the input can be obtained (Xu et al., 2005; Dyer et al., 2008; DeNeefe et al., 2008). Unfortunately, this approach is difficult to utilize because it requires multiple segmenters that behave differently on the same input.

In this paper, we describe a maximum entropy word segmentation model that is trained to assign high probability to possibly several segmentations of an input word. This model enables generation of diverse, accurate segmentation lattices from a single model that are appropriate for use in decoders that accept word lattices as input, such as Moses (Koehn et al., 2007). Since our model relies a small number of dense features, its parameters can be tuned using very small amounts of manually created *reference lattices*. Furthermore, since these parameters were chosen to have valid interpretation across a variety of languages, we find that the weights estimated for one apply quite well to another. We show that these lattices significantly improve translation quality when translating into English from three languages exhibiting productive compounding: German, Turkish, and Hungarian.

The paper is structured as follows. In the next sec-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 406–414,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

tion, we describe translation from segmentation lattices and give a motivating example, Section 3 describes our segmentation model and its tuning and how it is used to generate segmentation lattices, Section 5 presents experimental results, Section 6 reviews relevant related work, and in Section 7 we conclude and discuss future work.

2 Segmentation lattice translation

In this section we give a brief overview of lattice translation and then describe the characteristics of segmentation lattices that are appropriate for translation.

2.1 Lattice translation

Word lattices have been used to represent ambiguous input to machine translation systems for a variety of tasks, including translating automatic speech recognition transcriptions and translating from morphologically complex languages (Bertoldi et al., 2007; Dyer et al., 2008). The intuition behind using lattices in both approaches is to avoid the error propagation effects that are found when a one-best guess is used. By carrying a certain amount of uncertainty forward in the processing pipeline, information contained in the translation models can be leveraged to help resolve the upstream ambiguity. In our case, we want to propagate uncertainty about the proper segmentation of a compound forward to the decoder, which can use its full translation model to select proper segmentation for translation. Mathematically, this can be understood as follows: whereas the goal in conventional machine translation is to find the sentence $\hat{e}_1^I$ that maximizes $Pr(e_1^I|f_1^J)$, the lattice adds a latent variable, the path $\bar{f}$ from a designated start start to a designated goal state in the lattice $\mathcal{G}$:

$$\hat{e}_1^I = \arg\max_{e_1^I} Pr(e_1^I|\mathcal{G}) \quad (1)$$

$$= \arg\max_{e_1^I} \sum_{\bar{f}\in\mathcal{G}} Pr(e_1^I|\bar{f})Pr(\bar{f}|\mathcal{G}) \quad (2)$$

$$\approx \arg\max_{e_1^I} \max_{\bar{f}\in\mathcal{G}} Pr(e_1^I|\bar{f})Pr(\bar{f}|\mathcal{G}) \quad (3)$$

If the transduction formalism used is a synchronous probabilistic context free grammar or weighted finite state transducer, the search represented by equation (3) can be carried out efficiently using dynamic programming (Dyer et al., 2008).

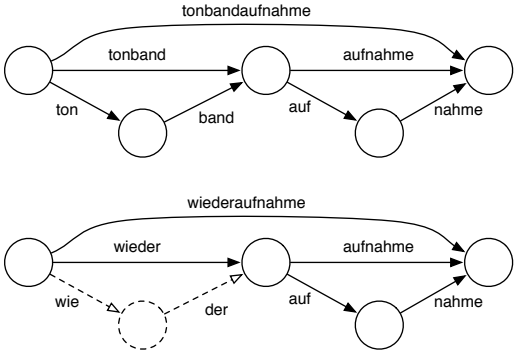


Figure 1: Segmentation lattice examples. The dotted structure indicates linguistically implausible segmentation that might be generated using dictionary-driven approaches.

2.2 Segmentation lattices

Figure 1 shows two lattices that encode the most linguistically plausible ways of segmenting two prototypical German compounds with compositional meanings. However, while these words are structurally quite similar, translating them into English would seem to require different amounts of segmentation. For example, the dictionary fragment shown in Table 1 illustrates that *tonbandaufnahme* can be rendered into English by following 3 different paths in the lattice, *ton*/audio *band*/tape *aufnahme*/recording, *tonband*/tape *aufnahme*/recording, and *tonbandaufnahme*/tape recording. In contrast, *wiederaufnahme* can only be translated correctly using the unsegmented form, even though in German the meaning of the full form is a composition of the meaning of the individual morphemes.[1]

It should be noted that phrase-based models can translate multiple words as a unit, and therefore capture non-compositional meaning. Thus, by default if the training data is processed such that, for example, *aufnahme*, in its sense of *recording*, is segmented into two words, then more paths in the lattices be-

[1]The English word *resumption* is likewise composed of two morphemes, the prefix *re-* and a kind of bound morpheme that never appears in other contexts (sometimes called a 'cranberry' morpheme), but the meaning of the whole is idiosyncratic enough that it cannot be called compositional.

German	English
auf	*on, up, in, at, ...*
aufnahme	*recording, entry*
band	*reel, tape, band*
der	*the, of the*
nahme	*took* (3P-SG-PST)
ton	*sound, audio, clay*
tonband	*tape, audio tape*
tonbandaufnahme	*tape recording*
wie	*how, like, as*
wieder	*again*
wiederaufnahme	*resumption*

Table 1: German-English dictionary fragment for words present in Figure 1.

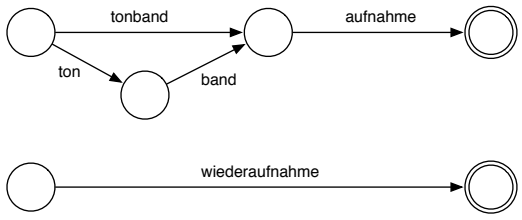


Figure 2: Manually created reference lattices for the two words from Figure 1. Although only a subset of all linguistically plausible segmentations, each path corresponds to a plausible segmentation for word-for-word German-English translation.

come plausible translations. However, using a strategy of "over segmentation" and relying on phrase models to learn the non-compositional translations has been shown to degrade translation quality significantly on several tasks (Xu et al., 2004; Habash and Sadat, 2006). We thus desire lattices containing as little oversegmentation as possible.

We have now have a concept of a "gold standard" segmentation lattice for translation: it should contain all linguistically motivated segmentations that also correspond to plausible word-for-word translations into English. Figure 2 shows an example of the reference lattice for the two words we just discussed. For the experiments in this paper, we generated a development and test set by randomly choosing 19 German newspaper articles, identifying all words greater than 6 characters is length, and segmenting each word so that the resulting units could be translated compositionally into English. This resulted in 489 training sentences corresponding to 564 paths for the dev set (which was drawn from 15 articles), and 279 words (302 paths) for the test set (drawn from the remaining 4 articles).

3 A maximum entropy segmentation model

We now turn to the problem of modeling word segmentation in a way that facilitates lattice construction. As a starting point, we consider the work of Koehn and Knight (2003) who observe that in most languages that exhibit compounding, the morphemes used to construct compounds frequently also appear as individual tokens. Based on this observation, they propose a model of word segmentation that splits compound words into pieces found in the dictionary based on a variety heuristic scoring criteria. While these models have been reasonably successful (Koehn et al., 2008), they are problematic for two reasons. First, there is no principled way to incorporate additional features (such as phonotactics) which might be useful to determining whether a word break should occur. Second, the heuristic scoring offers little insight into which segmentations should be included in a lattice.

We would like our model to consider a wide variety of segmentations of any word (including perhaps hypothesized morphemes that are not in the dictionary), to make use of a rich set of features, and to have a probabilistic interpretation of each hypothesized split (to incorporate into the downstream decoder). We decided to use the class of maximum entropy models, which are probabilistically sound, can make use of possibly many overlapping features, and can be trained efficiently (Berger et al., 1996). We thus define a model of the conditional probability distribution $Pr(s_1^N|w)$, where w is a surface form and s_1^N is the segmented form consisting of N segments as:

$$Pr(s_1^N|w) = \frac{\exp \sum_i \lambda_i h_i(s_1^N, w)}{\sum_{s'} \exp \sum_i \lambda_i h_i(s', w)} \quad (4)$$

To simplify inference and to make the lattice representation more natural, we only make use of local feature functions that depend on properties of each segment:

$$Pr(s_1^N|w) \propto \exp \sum_i \lambda_i \sum_j^N h_i(s_j, w) \quad (5)$$

3.1 From model to segmentation lattice

The segmentation model just introduced is equivalent to a lattice where each vertex corresponds to a particular coverage (in terms of letters consumed from left to right) of the input word. Since we only make use of local features, the number of vertices in a lattice for word w is $|w| - m$, where m is the minimum segment length permitted. In all experiments reported in this paper, we use $m = 3$. Each edge is labeled with a morpheme s (corresponding to the morpheme associated with characters delimited by the start and end nodes of the edge) as well as a weight, $\sum_i \lambda_i h_i(s, w)$. The cost of any path from the start to the goal vertex will be equal to the numerator in equation (4). The value of the denominator can be computed using the forward algorithm.

In most of our experiments, s will be identical to the substring of w that the edge is designated to cover. However, this is not a requirement. For example, German compounds frequently have so-called *Fugenelemente*, one or two characters that "glue together" the primary morphemes in a compound. Since we permit these characters to be deleted, then an edge where they are deleted will have fewer characters than the coverage indicated by the edge's starting and ending vertices.

3.2 Lattice pruning

Except for the minimum segment length restriction, our model defines probabilities for all segmentations of an input word, making the resulting segmentation lattices are quite large. Since large lattices are costly to deal with during translation (and may lead to worse translations because poor segmentations are passed to the decoder), we prune them using forward-backward pruning so as to contain just the highest probability paths (Sixtus and Ortmanns, 1999). This works by computing the score of the best path passing through every edge in the lattice using the forward-backward algorithm. By finding the best score overall, we can then prune edges using a threshold criterion; i.e., edges whose score is some factor α away from the global best edge score.

3.3 Maximum likelihood training

Our model defines a conditional probability distribution over virtually all segmentations of a word w. To train our model, we wish to maximize the likelihood of the segmentations contained in the reference lattices by moving probability mass away from the segmentations that are *not* in the reference lattice. Thus, we wish to minimize the following objective (which can be computed using the forward algorithm over the unpruned hypothesis lattices):

$$\mathcal{L} = -\log \sum_i \sum_{\mathbf{s} \in \mathcal{R}_i} p(\mathbf{s}|w_i) \quad (6)$$

The gradient with respect to the feature weights for a log linear model is simply:

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = \sum_i \mathbb{E}_{p(\mathbf{s}|w_i)}[h_k] - \mathbb{E}_{p(\mathbf{s}|w_i, \mathcal{R}_i)}[h_k] \quad (7)$$

To compute these values, the first expectation is computed using forward-backward inference over the full lattice. To compute the second expectation, the full lattice is intersected with the reference lattice $\mathcal{R}_i$, and then forward-backward inference is redone.[2] We use the standard quasi-Newtonian method L-BFGS to optimize the model (Liu et al., 1989). Training generally converged in only a few hundred iterations.

3.3.1 Training to minimize 1-best error

In some cases, such as when performing word alignment for translation model construction, lattices cannot be used easily. In these cases, a 1-best segmentation (which can be determined from the lattice using the Viterbi algorithm) may be desired. To train the parameters of the model for this condition (which is arguably slightly different from the lattice generation case we just considered), we used the minimum error training (MERT) algorithm on the segmentation lattices to find the parameters that minimized the error on our dev set (Macherey

[2]The second expectation corresponds to the empirical feature observations in a standard maximum entropy model. Because this is an expectation and not an invariant observation, the log likelihood function is not guaranteed to be concave and the objective surface may have local minima. However, experimentation revealed the optimization performance was largely invariant with respect to its starting point.

et al., 2008). The error function we used was WER (the minimum number of insertions, substitutions, and deletions along any path in the reference lattice, normalized by the length of this path). The WER on the held-out test set for a system tuned using MERT is 9.9%, compared to 11.1% for maximum likelihood training.

3.4 Features

We remark that since we did not have the resources to generate training data in all the languages we wished to generate segmentation lattices for, we have confined ourselves to features that we expect to be reasonably informative for a broad class of languages. A secondary advantage of this is that we used denser features than are often used in maximum entropy modeling, meaning that we could train our model with relatively less training data than might otherwise be required.

The features we used in our compound segmentation model for the experiments reported below are shown in Table 2. Building on the prior work that relied heavily on the frequency of the hypothesized constituent morphemes in a monolingual corpus, we included features that depend on this value, $f(s_i)$. $|s_i|$ refers to the number of letters in the ith hypothesized segment. Binary predicates evaluate to 1 when true and 0 otherwise. $f(s_i)$ is the frequency of the token s_i as an independent word in a monolingual corpus. $p(\#|s_{i1} \cdots s_{i4})$ is the probability of a word start preceding the letters $s_{i1} \cdots s_{i4}$. We found it beneficial to include a feature that was the probability of a certain string of characters beginning a word, for which we used a reverse 5-gram character model and predicted the word boundary given the first five letters of the hypothesized word split.[3] Since we did have expertise in German morphology, we did build a special German model. For this, we permitted the strings *s*, *n*, and *es* to be deleted between words. Each deletion fired a count feature (listed as *fugen* in the table). Analysis of errors indicated that the segmenter would periodically propose an incorrect segmentation where a single word could be divided into a word and a nonword consisting of common inflectional suffixes. To address this, an additional feature was added that fired when a proposed segment was one of a set $\mathcal{N}$ of 30 nonwords that we saw quite frequently. The weights shown in Table 2 are those learned by maximum likelihood training on models both with and without the special German features, which are indicated with †.

Feature	de-only	neutral
$^{\dagger}s_i \in \mathcal{N}$	-3.55	–
$f(s_i) > 0.005$	-3.13	-3.31
$f(s_i) > 0$	3.06	3.64
$\log p(\#\|s_{i1}s_{i2}s_{i3}s_{i4})$	-1.58	-2.11
segment penalty	1.18	2.04
$\|s_i\| \geq 12$	-0.9	-0.79
oov	-0.88	-1.09
†*fugen*	-0.76	–
$\|s_i\| \leq 4$	-0.66	-1.18
$\|s_i\| \leq 10,\ f(s_i) > 2^{-10}$	-0.51	-0.82
$\log f(s_i)$	-0.32	-0.36
$2^{-10} < f(s_i) < 0.005$	-0.26	-0.45

Table 2: Features and weights learned by maximum likelihood training, sorted by weight magnitude.

4 Model evalatuion

To give some sense of the performance of the model in terms of its ability to generate lattices independently of a translation task, we present precision and recall of segmentations for pruning parameters (cf. Section 3.2) ranging from $\alpha = 0$ to $\alpha = 5$. Precision measures the number of paths in the hypothesized lattice that correspond to paths in the reference lattice; recall measures the number of paths in the reference lattices that are found in the hypothesis lattice. Figure 3 shows the effect of manipulating the density parameter on the precision and recall of the German lattices. Note that very high recall is possible; however, the German-only features have a significant impact, especially on recall, because the reference lattices include paths where *Fugenelemente* have been deleted.

5 Translation experiments

We now review experiments using segmentation lattices produced by the segmentation model we just introduced in German-English, Hungarian-English,

[3] In general, this helped avoid situations where a word may be segemented into a frequent word and then a non-word string of characters since the non-word typically violated the phonotactics of the language in some way.

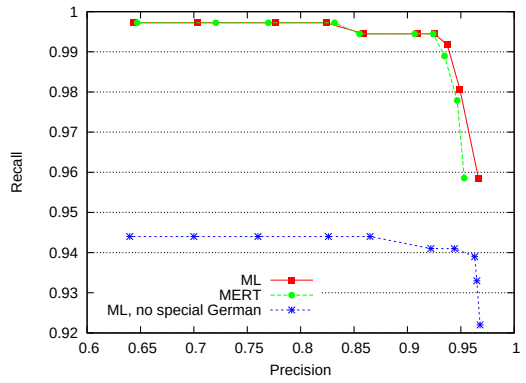


Figure 3: The effect of the lattice density parameter on precision and recall.

and Turkish-English translation tasks and then show results elucidating the effect of the lattice density parameter. We begin with a description of our MT system.

5.1 Data preparation and system description

For all experiments, we used a 5-gram English language model trained on the AFP and Xinua portions of the Gigaword v3 corpus (Graff et al., 2007) with modified Kneser-Ney smoothing (Kneser and Ney, 1995). The training, development, and test data for German-English and Hungarian-English systems used were distributed as part of the 2009 EACL Workshop on Machine Translation,[4] and the Turkish-English data corresponds to the training and test sets used in the work of Oflazer and Durgar El-Kahlout (2007). Corpus statistics for all language pairs are summarized in Table 3. We note that in all language pairs, the 1BEST segmentation variant of the training data results in a significant reduction in types.

Word alignment was carried out by running Giza++ implementation of IBM Model 4 initialized with 5 iterations of Model 1, 5 of the HMM aligner, and 3 iterations of Model 4 (Och and Ney, 2003) in both directions and then symmetrizing using the `grow-diag-final-and` heuristic (Koehn et al., 2003). For each language pair, the corpus was aligned twice, once in its non-segmented variant and once using the single-best segmentation variant.

For translation, we used a bottom-up parsing decoder that uses cube pruning to intersect the language model with the target side of the synchronous grammar. The grammar rules were extracted from the word aligned parallel corpus and scored as described in Chiang (2007). The features used by the decoder were the English language model log probability, $\log f(\bar{e}|\bar{f})$, the 'lexical translation' log probabilities in both directions (Koehn et al., 2003), and a word count feature. For the lattice systems, we also included the unnormalized $\log p(\bar{f}|\mathcal{G})$, as it is defined in Section 3, as well as an *input* word count feature. The feature weights were tuned on a held-out development set so as to maximize an equally weighted linear combination of BLEU and 1-TER (Papineni et al., 2002; Snover et al., 2006) using the minimum error training algorithm on a packed forest representation of the decoder's hypothesis space (Macherey et al., 2008). The weights were independently optimized for each language pair and each experimental condition.

5.2 Segmentation lattice results

In this section, we report the results of an experiment to see if the compound lattices constructed using our maximum entropy model yield better translations than either an unsegmented baseline or a baseline consisting of a single-best segmentation.

For each language pair, we define three conditions: BASELINE, 1BEST, and LATTICE. In the BASELINE condition, a lowercased and tokenized (but not segmented) version of the test data is translated using the grammar derived from a non-segmented training data. In the 1BEST condition, the single best segmentation $\hat{s}_1^N$ that maximizes $Pr(s_1^N|w)$ is chosen for each word using the MERT-trained model (the German model for German, and the language-neutral model for Hungarian and Turkish). This variant is translated using a grammar induced from a parallel corpus that has also been segmented according to the same decision rule. In the LATTICE condition, we constructed segmentation lattices using the technique described in Section 3.1. For all languages pairs, we used $d = 2$ as the pruning density parameter (which corresponds to the highest F-score on the held out test set). Additionally, if the unsegmented form of the word was removed from the lattice during pruning, it was restored to the lattice with zero weight.

Table 4 summarizes the results of the translation

[4] http://www.statmt.org/wmt09

	f-tokens	f-types	e-tokens.	e-types
DE-BASELINE	38M	307k	40M	96k
DE-1BEST	40M	136k	”	”
HU-BASELINE	25M	646k	29M	158k
HU-1BEST	27M	334k	”	”
TR-BASELINE	1.0M	56k	1.3M	23k
TR-1BEST	1.1M	41k	”	”

Table 3: Training corpus statistics.

	BLEU	TER
DE-BASELINE	21.0	60.6
DE-1BEST	20.7	60.1
DE-LATTICE	**21.6**	**59.8**
HU-BASELINE	11.0	71.1
HU-1BEST	10.7	70.4
HU-LATTICE	**12.3**	**69.1**
TR-BASELINE	26.9	61.0
TR-1BEST	27.8	61.2
TR-LATTICE	**28.7**	**59.6**

Table 4: Translation results for German (DE)-English, Hungarian (HU)-English, and Turkish (TR)-English. Scores were computed using a single reference and are case insensitive.

experiments comparing the three input variants. For all language pairs, we see significant improvements in both BLEU and TER when segmentation lattices are used.[5] Additionally, we also confirmed previous findings that showed that when a large amount of training data is available, moving to a one-best segmentation does not yield substantial improvements (Yang and Kirchhoff, 2006). Perhaps most surprisingly, the improvements observed when using lattices with the Hungarian and Turkish systems were *larger* than the corresponding improvement in the German system, but German was the only language for which we had segmentation training data. The smaller effect in German is probably due to there being more in-domain training data in the German system than in the (otherwise comparably sized) Hungarian system.

Targeted analysis of the translation output shows that while both the 1BEST and LATTICE systems generally produce adequate translations of compound words that are out of vocabulary in the BASELINE system, the LATTICE system performs better since it recovers from infelicitous splits that the one-best segmenter makes. For example, one class of error we frequently observe is that the one-best segmenter splits an OOV proper name into two pieces when a portion of the name corresponds to a known word in the source language (e.g. *tom tancredo→tom tan credo* which is then translated as *tom tan belief*).[6]

5.3 The effect of the density parameter

Figure 4 shows the effect of manipulating the density parameter (cf. Section 3.2) on the performance and decoding time of the Turkish-English translation system. It further confirms the hypothesis that increased diversity of segmentations encoded in a segmentation lattice can improve translation performance; however, it also shows that once the density becomes too great, and too many implausible segmentations are included in the lattice, translation quality will be harmed.

6 Related work

Aside from improving the vocabulary coverage of machine translation systems (Koehn et al., 2008; Yang and Kirchhoff, 2006; Habash and Sadat, 2006), compound word segmentation (also referred to as *decompounding*) has been shown to be helpful in a variety of NLP tasks including mono- and

[5]Using bootstrap resampling (Koehn, 2004), the improvements in BLEU, TER, as well as the linear combination used in tuning are statistically significant at at least $p < .05$.

[6]We note that our maximum entropy segmentation model could easily address this problem by incorporating information about whether a word is likely to be a named entity as a feature.

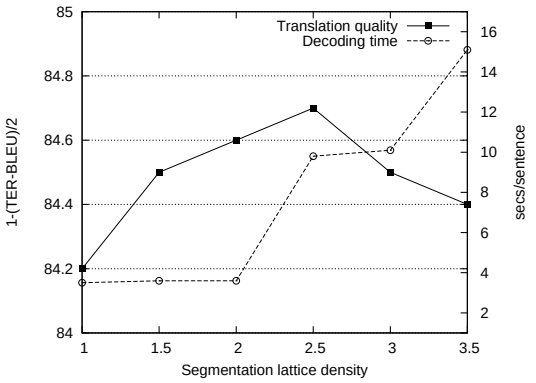


Figure 4: The effect of the lattice density parameter on translation quality and decoding time.

crosslingual IR (Airio, 2006) and speech recognition (Hessen and Jong, 2003). A number of researchers have demonstrated the value of using lattices to encode segmentation alternatives as input to a machine translation system (Dyer et al., 2008; DeNeefe et al., 2008; Xu et al., 2004), but this is the first work to do so using a single segmentation model. Another strand of inquiry that is closely related is the work on adjusting the source language segmentation to match the granularity of the target language as a way of improving translation. The approaches suggested thus far have been mostly of a heuristic nature tailored to Chinese-English translation (Bai et al., 2008; Ma et al., 2007).

7 Conclusions and future work

In this paper, we have presented a maximum entropy model for compound word segmentation and used it to generate segmentation lattices for input into a statistical machine translation system. These segmentation lattices improve translation quality (over an already strong baseline) in three typologically distinct languages (German, Hungarian, Turkish) when translating into English. Previous approaches to generating segmentation lattices have been quite laborious, relying either on the existence of multiple segmenters (Dyer et al., 2008; Xu et al., 2005) or hand-crafted rules (DeNeefe et al., 2008). Although the segmentation model we propose is discriminative, we have shown that it can be trained using a minimal amount of annotated training data. Furthermore, when even this minimal data cannot be acquired for a particular language (as was the situation we faced with Hungarian and Turkish), we have demonstrated that the parameters obtained in one language work surprisingly well for others. Thus, with virtually no cost, this model can be used with a variety of diverse languages.

While these results are already quite satisfying, there are a number of compelling extensions to this work that we intend to explore in the future. First, unsupervised segmentation approaches offer a very compelling alternative to the manually crafted segmentation lattices that we created. Recent work suggests that unsupervised segmentation of inflectional affixal morphology works quite well (Poon et al., 2009), and extending this work to compounding morphology should be feasible, obviating the need for expensive hand-crafted reference lattices. Second, incorporating target language information into a segmentation model holds considerable promise for inducing more effective translation models that perform especially well for segmentation lattice inputs.

Acknowledgments

Special thanks to Kemal Oflazar and Reyyan Yeniterzi of Sabancı University for providing the Turkish-English corpus and to Philip Resnik, Adam Lopez, Trevor Cohn, and especially Phil Blunsom for their helpful suggestions. This research was supported by the Army Research Laboratory. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the sponsors.

References

Eija Airio. 2006. Word normalization and decompounding in mono- and bilingual IR. *Information Retrieval*, 9:249–271.

Ming-Hong Bai, Keh-Jiann Chen, and Jason S. Chang. 2008. Improving word alignment by adjusting Chinese word segmentation. In *Proceedings of the Third International Joint Conference on Natural Language Processing*.

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

N. Bertoldi, R. Zens, and M. Federico. 2007. Speech

translation by confusion network decoding. In *Proceeding of ICASSP 2007*, Honolulu, Hawaii, April.

Pi-Chuan Chang, Dan Jurafsky, and Christopher D. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, Prague, Czech Republic, June.

D. Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

S. DeNeefe, U. Hermjakob, and K. Knight. 2008. Overcoming vocabulary sparsity in mt using lattices. In *Proceedings of AMTA*, Honolulu, HI.

C. Dyer, S. Muresan, and P. Resnik. 2008. Generalizing word lattice translation. In *Proceedings of HLT-ACL*.

D. Graff, J. Kong, K. Chen, and K. Maeda. 2007. English gigaword third edition.

N. Habash and F. Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *Proc. of NAACL*, New York.

Arjan Van Hessen and Franciska De Jong. 2003. Compound decomposition in dutch large vocabulary speech recognition. In *Proceedings of Eurospeech 2003, Geneve*, pages 225–228.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of IEEE Internation Conference on Acoustics, Speech, and Signal Processing*, pages 181–184.

P. Koehn and K. Knight. 2003. Empirical methods for compound splitting. In *Proc. of the EACL 2003*.

P. Koehn, F.J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL 2003*, pages 48–54, Morristown, NJ, USA. Association for Computational Linguistics.

P. Koehn, H. Hoang, A. Birch Mayne, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computation Linguistics (ACL), Demonstration Session*, pages 177–180, June.

Philipp Koehn, Abhishek Arun, and Hieu Hoang. 2008. Towards better machine translation quality for the German-English language pairs. In *ACL Workshop on Statistical Machine Translation*.

P. Koehn. 2004. Statistical signficiance tests for machine translation evluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 388–395.

Dong C. Liu, Jorge Nocedal, Dong C. Liu, and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming B*, 45(3):503–528.

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

Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of EMNLP*, Honolulu, HI.

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

Kemal Oflazer and Ilknur Durgar El-Kahlout. 2007. Exploring different representational units in English-to-Turkish statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 25–32, Prague, Czech Republic, June. Association for Computational Linguistics.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 311–318.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proc. of NAACL 2009*.

S. Sixtus and S. Ortmanns. 1999. High quality word graphs using forward-backward pruning. In *Proceedings of ICASSP*, Phoenix, AZ.

Matthew Snover, Bonnie J. Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.

J. Xu, R. Zens, and H. Ney. 2004. Do we need Chinese word segmentation for statistical machine translation? In *Proceedings of the Third SIGHAN Workshop on Chinese Language Learning*, pages 122–128, Barcelona, Spain.

J. Xu, E. Matusov, R. Zens, and H. Ney. 2005. Integrated Chinese word segmentation in statistical machine translation. In *Proc. of IWSLT 2005*, Pittsburgh.

M. Yang and K. Kirchhoff. 2006. Phrase-based backoff models for machine translation of highly inflected languages. In *Proceedings of the EACL 2006*, pages 41–48.

Active Learning for Statistical Phrase-based Machine Translation*

Gholamreza Haffari and **Maxim Roy** and **Anoop Sarkar**
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
{ghaffar1,maximr,anoop}@cs.sfu.ca

Abstract

Statistical machine translation (SMT) models need large bilingual corpora for training, which are unavailable for some language pairs. This paper provides the first serious experimental study of active learning for SMT. We use active learning to improve the quality of a phrase-based SMT system, and show significant improvements in translation compared to a random sentence selection baseline, when test and training data are taken from the same or different domains. Experimental results are shown in a simulated setting using three language pairs, and in a realistic situation for Bangla-English, a language pair with limited translation resources.

1 Introduction

Statistical machine translation (SMT) systems have made great strides in translation quality. However, high quality translation output is dependent on the availability of massive amounts of parallel text in the source and target language. However, there are a large number of languages that are considered "low-density", either because the population speaking the language is not very large, or even if millions of people speak the language, insufficient amounts of parallel text are available in that language.

A statistical translation system can be improved or adapted by incorporating new training data in the form of parallel text. In this paper, we propose several novel *active learning* (AL) strategies for statistical machine translation in order to attack this problem. Conventional techniques for AL of classifiers are problematic in the SMT setting. Selective sampling of sentences for AL may lead to a parallel corpus where each sentence does not share any phrase pairs with the others. Thus, new sentences cannot be translated since we lack evidence for how phrase pairs combine to form novel translations. In this paper, we take the approach of exploration vs. exploitation: where in some cases we pick sentences that are not entirely novel to improve translation statistics, while also injecting novel translation pairs to improve coverage.

There may be evidence to show that AL is useful even when we have massive amounts of parallel training data. (Turchi et al., 2008) presents a comprehensive learning curve analysis of a phrase-based SMT system, and one of the conclusions they draw is, "The first obvious approach is an effort to identify or produce data sets on demand (active learning, where the learning system can request translations of specific sentences, to satisfy its information needs)."

Despite the promise of active learning for SMT there has been very little experimental work published on this issue (see Sec. 5). In this paper, we make several novel contributions to the area of active learning for SMT:

• We use a novel framework for AL, which to our knowledge has not been used in AL experiments before. We assume a small amount of parallel text and a large amount of monolingual source language text. Using these resources, we create a large noisy parallel text which we then iteratively improve using small injections of human translations.

• We provide many useful and novel features useful for AL in SMT. In translation, we can leverage a whole new set of features that were out of reach for classification systems: we devise features that look at the source language, but also devise features that make an estimate of the potential utility of translations from the source, e.g. phrase pairs that could be extracted.

• We show that AL can be useful in domain adaptation. We provide the first experimental evidence in SMT that active learning can be used to inject care-

*We would like to thank Chris Callison-Burch for fruitful discussions. This research was partially supported by NSERC, Canada (RGPIN: 264905) and by an IBM Faculty Award to the third author.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 415–423,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

fully selected translations in order to improve SMT output in a new domain.

• We compare our proposed features to a random selection baseline in a simulated setting for three language pairs. We also use a realistic setting: using human expert annotations in our AL system we create an improved SMT system to translate from Bangla to English, a language pair with very few resources.

2 An Active Learning Framework for SMT

Starting from an SMT model trained initially on bilingual data, the problem is to minimize the human effort in translating new sentences which will be added to the training data to make the *retrained* SMT model achieves a certain level of performance. Thus, given a bitext $L := \{(\mathbf{f}_i, \mathbf{e}_i)\}$ and a monolingual source text $U := \{\mathbf{f}_j\}$, the goal is to select a subset of highly informative sentences from U to present to a human expert for translation. Highly informative sentences are those which, together with their translations, help the retrained SMT system *quickly* reach a certain level of translation quality. This learning scenario is known as active learning with Selective Sampling (Cohn et al., 1994).

Algorithm 1 describes the experimental setup we propose for active learning. We train our initial MT system on the bilingual corpus L, and use it to translate *all* monolingual sentences in U. We denote sentences in U together with their translations as U^+ (line 4 of Algorithm 1). Then we retrain the SMT system on $L \cup U^+$ and use the resulting model to decode the test set. Afterwards, we select and remove a subset of highly informative sentences from U, and add those sentences together with their human-provided translations to L. This process is continued iteratively until a certain level of translation quality, which in our case is measured by the BLEU score, is met. In the baseline, against which we compare our sentence selection methods, the sentences are chosen *randomly*.

When (re-)training the model, two phrase tables are learned: one from L and the other one from U^+. The phrase table obtained from U^+ is added as a new feature function in the log-linear translation model. The alternative is to ignore U^+ as in a conventional AL setting, however, in our experiments we have found that using more bilingual data, even noisy data, results in better translations.

Algorithm 1 AL-SMT

1: Given bilingual corpus L, and monolingual corpus U.
2: $M_{F \to E} = \textbf{train}(L, \emptyset)$
3: **for** $t = 1, 2, ...$ **do**
4: $\quad U^+ = \textbf{translate}(U, M_{F \to E})$
5: $\quad$ Select k sentence pairs from U^+, and ask a human for their *true* translations.
6: $\quad$ Remove the k sentences from U, and add the k sentence pairs (translated by human) to L
7: $\quad M_{F \to E} = \textbf{train}(L, U^+)$
8: $\quad$ Monitor the performance on the test set T
9: **end for**

Phrase tables from U^+ will get a 0 score in minimum error rate training if they are not useful, so our method is more general. Also, this method has been shown empirically to be more effective (Ueffing et al., 2007b) than (1) using the weighted combination of the two phrase tables from L and U^+, or (2) combining the two sets of data and training from the bitext $L \cup U^+$.

The setup in Algorithm 1 helps us to investigate how to maximally take advantage of human effort (for sentence translation) when learning an SMT model from the available data, that includes bilingual and monolingual text.

3 Sentence Selection Strategies

Our sentence selection strategies can be divided into two categories: (1) those which are independent of the target language and just look into the source language, and (2) those which also take into account the target language. From the description of the methods, it will be clear to which category they belong to. We will see in Sec. 4 that the most promising sentence selection strategies belong to the second category.

3.1 The Utility of Translation Units

Phrases are basic units of translation in phrase-based SMT models. The phrases potentially extracted from a sentence indicate its informativeness. The more new phrases a sentence can offer, the more informative it is. Additionally phrase translation probabilities need to be estimated accurately, which means sentences that contain rare phrases are also informative. When selecting new sentences for hu-

man translation, we need to pay attention to this tradeoff between *exploration* and *exploitation*, i.e. selecting sentences to discover new phrases vs estimating accurately the phrase translation probabilities. A similar argument can be made that emphasizes the importance of words rather than phrases for any SMT model. Also we should take into account that smoothing is a means for accurate estimation of translation probabilities when events are rare. In our work, we focus on methods that effectively expand the lexicon or set of phrases of the model.

3.1.1 Phrases (Geom-Phrase, Arith-Phrase)[1]

The more frequent a phrase is in the *unlabeled* data, the more important it is to know its translation; since it is more likely to occur in the test data (especially when the test data is in-domain with respect to unlabeled data). The more frequent a phrase is in the *labeled* data, the more unimportant it is; since probably we have observed most of its translations.

Based on the above observations, we measure the importance score of a sentence as:

$$\phi_g^p(s) := \Big[\prod_{x \in X_s^p} \frac{P(x|U)}{P(x|L)} \Big]^{\frac{1}{|X_s^p|}} \quad (1)$$

where X_s^p is the set of possible phrases that sentence s can offer, and $P(x|\mathcal{D})$ is the probability of observing x in the data $\mathcal{D}$: $P(x|\mathcal{D}) = \frac{Count(x)+\epsilon}{\sum_{x \in X_\mathcal{D}^p} Count(x)+\epsilon}$. The score (1) is the averaged *probability ratio* of the set of candidate phrases, i.e. the probability of the candidate phrases under a probabilistic phrase model based on U divided by that based on L. In addition to the geometric average in (1), we may also consider the arithmetic average score:

$$\phi_a^p(s) := \frac{1}{|X_s^p|} \sum_{x \in X_s^p} \frac{P(x|U)}{P(x|L)} \quad (2)$$

Note that (1) can be re-written as $\frac{1}{|X_s^p|} \sum_{x \in X_s^p} \log \frac{P(x|U)}{P(x|L)}$ in the logarithm space, which is similar to (2) with the difference of additional $\log$.

In parallel data L, phrases are the ones which are extracted by the usual phrase extraction algorithm; but what are the candidate phrases in the unlabeled data? Considering the k-best list of translations can tell us the possible phrases the input sentence may offer. For each translation, we have access to the phrases used by the decoder to produce that output. However, there may be islands of out-of-vocabulary (OOV) words that were not in the phrase table and not translated by the decoder as a phrase. We group together such groups of OOV words to form an OOV phrase. The set of possible phrases we extract from the decoder output contain those coming from the phrase table (from labeled data L) and those coming from OOVs. OOV phrases are also used in our computation, where $P(x \mid L)$ for an OOV phrase x is the uniform probability over all OOV phrases.

3.1.2 n-grams (Geom n-gram, Arith n-gram)

As an alternative to phrases, we consider n-grams as basic units of generalization. The resulting score is the weighted combination of the n-gram based scores:

$$\phi_g^N(s) := \sum_{n=1}^{N} \frac{w_n}{|X_s^n|} \sum_{x \in X_s^n} \log \frac{P(x|U,n)}{P(x|L,n)} \quad (3)$$

where X_s^n denotes n-grams in the sentence s, and $P(x|\mathcal{D},n)$ is the probability of x in the set of n-grams in $\mathcal{D}$. The weights w_n adjust the importance of the scores of n-grams with different lengths. In addition to taking geometric average, we also consider the arithmetic average:

$$\phi_a^N(s) := \sum_{n=1}^{N} \frac{w_n}{|X_s^n|} \sum_{x \in X_s^n} \frac{P(x|U,n)}{P(x|L,n)} \quad (4)$$

As a special case when $N = 1$, the score motivates selecting sentences which increase the number of unique words with new words appearing with higher frequency in U than L.

3.2 Similarity to the Bilingual Training Data (Similarity)

The simplest way to expand the lexicon set is to choose sentences from U which are as dissimilar as possible to L. We measure the similarity using weighted n-gram coverage (Ueffing et al., 2007b).

3.3 Confidence of Translations (Confidence)

The decoder produces an output translation $\mathbf{e}$ using the probability $p(\mathbf{e} \mid \mathbf{f})$. This probability can be

[1]The names in the parentheses are short names used to identify the method in the experimental results.

treated as a confidence score for the translation. To make the confidence score for sentences with different lengths comparable, we normalize using the sentence length (Ueffing et al., 2007b).

3.4 Feature Combination (Combined)

The idea is to take into account the information from several simpler methods, e.g. those mentioned in Sec. 3.1–3.3, when producing the final ranking of sentences. We can either merge the output rankings of those simpler models[2], or use the scores generated by them as input *features* for a higher level ranking model. We use a linear model:

$$F(s) = \sum_k \alpha_k \phi_k(s) \quad (5)$$

where α_k are the model parameters, and $\phi_k(.)$ are the feature functions from Sections 3.1–3.3, e.g. confidence score, similarity to L, and score for the utility of translation units. Using 20K of Spanish unlabeled text we compared the r^2 correlation coefficient between each of these scores which, apart from the arithmetic and geometric versions of the same score, showed low correlation. And so the information they provide should be complementary to each other.

We train the parameters in (5) using two bilingual development sets dev1 and dev2, the sentences in dev1 can be ranked with respect to the amount by which each particular sentence improves the BLEU score of the retrained[3] SMT model on dev2. Having this ranking, we look for the weight vector which produces the same ordering of sentences. As an alternative to this method (or its computationally demanding generalization in which instead of a single sentence, several sets of sentences of size k are selected and ranked) we use a hill climbing search on the surface of dev2's BLEU score. For a fixed value of the weight vector, dev1 sentences are ranked and then the top-k output is selected and the amount of improvement the retrained SMT system gives on dev2's BLEU score is measured. Starting from a random initial value for α_k's, we improve one dimension at a time and traverse the discrete grid placed on the values of the weight vector. Starting with a coarse grid, we make it finer when we get stuck in local optima during hill climbing.

3.5 Hierarchical Adaptive Sampling (HAS)

(Dasgupta and Hsu, 2008) propose a technique for sample selection that, under certain settings, is guaranteed to be no worse than random sampling. Their method exploits the cluster structure (if there is any) in the unlabeled data. Ideally, querying the label of only one of the data points in a cluster would be enough to determine the label of the other data points in that cluster. Their method requires that the data set is provided in the form of a tree representing a hierarchical clustering of the data. In AL for SMT, such a unique clustering of the unlabeled data would be inappropriate or ad-hoc. For this reason, we present a new algorithm inspired by the rationale provided in (Dasgupta and Hsu, 2008) that can be used in our setting, where we construct a tree-based partition of the data dynamically[4]. This dynamic tree construction allows us to extend the HAS algorithm from classifiers to the SMT task.

The algorithm adaptively samples sentences from U while building a hierarchical clustering of the sentences in U (see Fig. 1 and Algorithm 2). At any iteration, first we retrain the SMT model and translate all monolingual sentences. At this point one monolingual set of sentences represented by one of the tree leaves is chosen for further partitioning: a leaf H is chosen which has the lowest average decoder confidence score for its sentence translations. We then rank all sentences in H based on their similarity to L and put the top $\alpha|H|$ sentences in H_1 and the rest in H_2. To select K sentences, we randomly sample βK sentences from H_1 and $(1-\beta)K$ sentences from H_2 and ask a human for their translations.

3.6 Reverse Model (Reverse)

While a translation system $M_{F\to E}$ is built from language F to language E, we also build a translation system in the reverse direction $M_{E\to F}$. To measure how informative a monolingual sentence $\mathbf{f}$ is, we translate it to English by $M_{F\to E}$ and then project

[2]To see how different rankings can be combined, see (Reichart et al., 2008) which proposes this for multi-task AL.

[3]Here the retrained SMT model is the one learned by adding a particular sentence from dev1 into L.

[4]The dynamic nature of the hierarchy comes from two factors: (1) selecting a leaf node for splitting, and (2) splitting a leaf node based on its similarity to the growing L.

Algorithm 2 Hierarchical-Adaptive-Sampling

$M_{F\to E} = \textbf{train}(L, \emptyset)$
Initialize the tree T by setting its root to U
$v :=$ root(T)
for $t = 1, 2, ...$ **do**
  // rank and split sentence in v
  $X_1, X_2 :=$ Partition(L, v, α)
  // randomly sample and remove sents from X_i
  $Y_1, Y_2 :=$ Sampling(X_1, X_2, β)
  // make X_i children of node v in the tree T
  $T :=$ UpdateTree(X_1, X_2, v, T)
  // Y_i^+ has sents in Y_i together with human trans
  $L := L \cup Y_1^+ \cup Y_2^+$
  $M_{F\to E} = \textbf{train}(L, U)$
  for all leaves $l \in T$ **do**
    $Z[l] :=$ Average normalized confidence scores of sentence translations in l
  end for
  $v :=$ BestLeaf(T, Z)
  Monitor the performance on the test set
end for

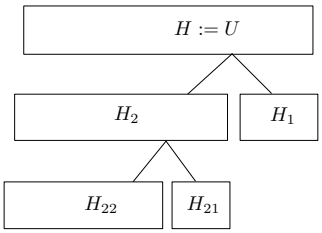


Figure 1: Adaptively sampling the sentences while constructing a hierarchical clustering of U.

the translation back to French using $M_{E\to F}$. Denote this *reconstructed* version of the original French sentence by $\tilde{\mathbf{f}}$. Comparing $\mathbf{f}$ with $\tilde{\mathbf{f}}$ using BLEU (or other measures) can tell us how much information has been lost due to our direct and/or reverse translation systems. The sentences with higher information loss are selected for translation by a human.

4 Experiments

The SMT system we applied in our experiments is PORTAGE (Ueffing et al., 2007a). The models (or features) which are employed by the decoder are: (a) one or several phrase table(s), which model the translation direction $p(\mathbf{f} \mid \mathbf{e})$, (b) one or several n-gram language model(s) trained with the SRILM toolkit (Stolcke, 2002); in the experiments reported here, we used 4-gram models on the NIST data, and a trigram model on EuroParl, (c) a distortion model which assigns a penalty based on the number of source words which are skipped when generating a new target phrase, and (d) a word penalty. These different models are combined log-linearly. Their weights are optimized w.r.t. BLEU score using the algorithm described in (Och, 2003). This is done on a development corpus which we will call dev1 in this paper.

corpus	language	use	sentences
EuroParl	Fr,Ge,Sp	in-dom L	5K
		in-dom U	20K
		in-dom dev	2K
		in-dom test	2K
See Sec. 4.2	Bangla	in-dom L	11K
		in-dom U	20K
		in-dom dev	450
		in-dom test	1K
Hansards	Fr	out-dom L	5K

Table 1: Specification of different data sets we will use in experiments. The target language is English in the bilingual sets, and the source languages are either French (Fr), German (Ge), Spanish (Sp), or Bangla.

The weight vectors in n-gram and similarity methods are set to $(.15, .2, .3, .35)$ to emphasize longer n-grams. We set $\alpha = \beta = .35$ for HAS, and use the 100-best list of translations when identifying candidate phrases while setting the maximum phrase length to 10. We set $\epsilon = .5$ to smooth probabilities when computing scores based on translation units.

4.1 Simulated Low Density Language Pairs

We use three language pairs (French-English, German-English, Spanish-English) to compare all of the proposed sentence selection strategies in a simulated AL setting. The training data comes from EuroParl corpus as distributed for the shared task in the NAACL 2006 workshop on statistical machine translation (WSMT06). For each language pair, the first 5K sentences from its bilingual corpus constitute L, and the next 20K sentences serve as U where the target side translation is ignored. The size of L was taken to be 5K in order to be close to a realistic setting in SMT. We use the first 2K sentences from the test sets provided for WSMT06, which are in-domain, as our test sets. The corpus statistics are summarized in Table 1. The results are shown in Fig. 2. After building the initial MT systems, we se-

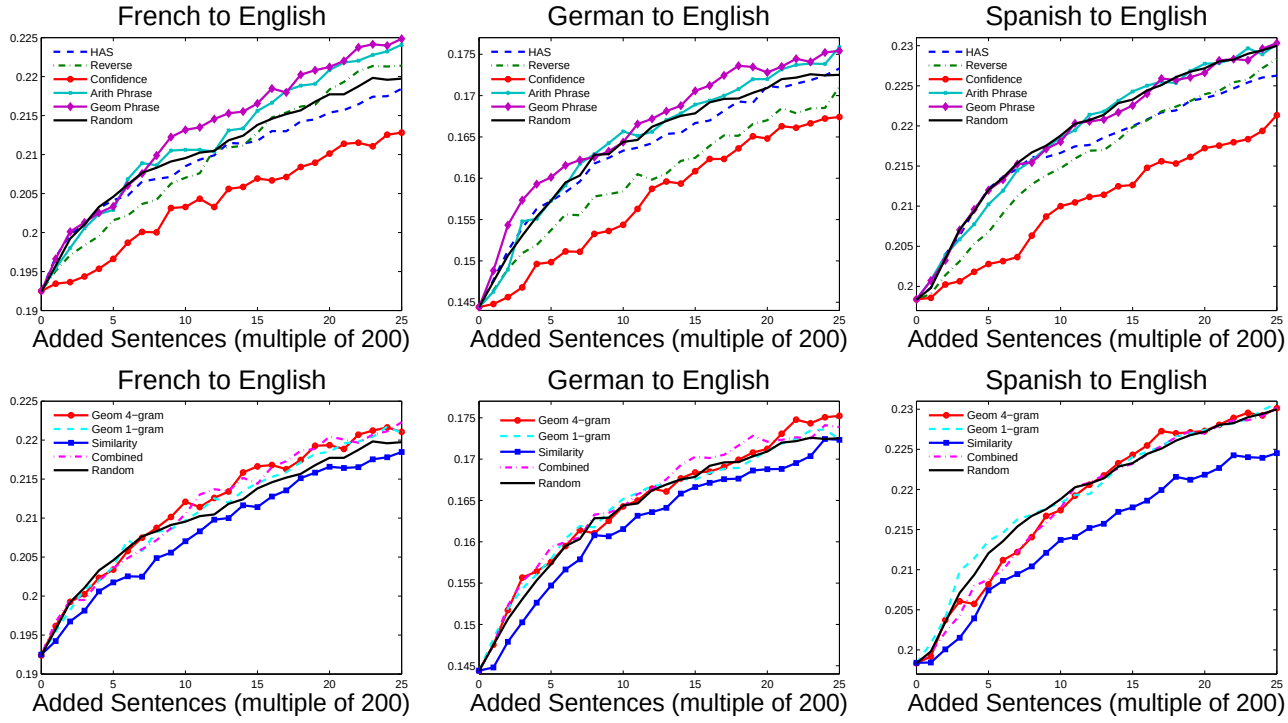


Figure 2: BLEU scores for different sentence selection strategies per iteration of the AL algorithm. Plots at the top show the performance of sentence selection methods which depend on the target language in addition to the source language (hierarchical adaptive sampling, reverse model, decoder confidence, average and geometric phrase-based score), and plots at the bottom show methods which are independent of the target language (geometric 4-gram and 1-gram, similarity to L, and random sentence selection baseline).

lect and remove 200 sentence from U in each iteration and add them together with translations to L for 25 iterations. Each experiment which involves randomness, such as random sentence selection baseline and HAS, is averaged over three independent runs. Selecting sentences based on the phrase-based utility score outperforms the strong random sentence selection baseline and other methods (Table 2). Decoder confidence performs poorly as a criterion for sentence selection in this setting, and HAS which is built on top of confidence and similarity scores outperforms both of them. Although choosing sentences based on their n-gram score ignores the relationship between source and target languages, this methods outperforms random sentence selection.

4.2 Realistic Low Density Language Pair

We apply active learning to the Bangla-English machine translation task. Bangla is the official language of Bangladesh and second most spoken language in India. It has more than 200 million speakers around the world. However, Bangla has few available language resources, and lacks resources for machine translation. In our experiments, we use training data provided by the Linguistic Data Consortium[5] containing $\sim$11k sentences. It contains newswire text from the BBC Asian Network and some other South Asian news websites. A bilingual Bangla-English dictionary collected from different websites was also used as part of the training set which contains around 85k words. Our monolingual corpus[6] is built by collecting text from the *Prothom Alo* newspaper, and contains all the news available for the year of 2005 – including magazines and periodicals. The corpus has 18,067,470 word tokens and 386,639 word types. For our language model we used data from the English section of EuroParl. The

[5]LDC Catalog No.: LDC2008E29.

[6]Provided by the Center for Research on Bangla Language Processing, BRAC University, Bangladesh.

development set used to optimize the model weights in the decoder, and test set used for evaluation was taken from the same LDC corpus mentioned above.

We applied our active learning framework to the problem of creating a larger Bangla-English parallel text resource. The second author is a native speaker of Bangla and participated in the active learning loop, translating 100 sentences in each iteration. We compared a smaller number of alternative methods to keep the annotation cost down. The results are shown in Fig. 3. Unlike the simulated setting, in this realistic setting for AL, adding more human translation does not always result in better translation performance[7]. Geom 4-gram and Geom phrase are the features that prove most useful in extracting useful sentences for the human expert to translate.

4.3 Domain Adaptation

In this section, we investigate the behavior of the proposed methods when unlabeled data U and test data T are in-domain and parallel training text L is out-of-domain.

We report experiments for French to English translation task where T and development sets are the same as those in section 4.1 but the bilingual training data come from Hansards[8] corpus. The domain is similar to EuroParl, but the vocabulary is very different. The results are shown in Fig. 4, and summarized in Table 3. As expected, unigram based sentence selection performs well in this scenario since it quickly expands the lexicon set of the bilingual data in an effective manner (Fig 5). By ignoring sentences for which the translations are already known based on L, it does not waste resources. On the other hand, it raises the importance of high frequency words in U. Interestingly, decoder confidence is also a good criterion for sentence selection in this particular case.

[7]This is likely due to the fact that the translator in the AL loop was not the same as the original translator for the labeled data.

[8]The transcription of official records of the Canadian Parliament as distributed at http://www.isi.edu/natural-language/download/hansard

Lang. Pair	Geom Phrase bleu%	per%	wer%	Random (baseline) bleu%	per%	wer%
Fr-En	22.49	27.99	38.45	21.97	28.31	38.80
Gr-En	17.54	31.51	44.28	17.25	31.63	44.41
Sp-En	23.03	28.86	39.17	23.00	28.97	39.21

Table 2: Phrase-based utility selection is compared with random sentence selection baseline with respect to BLEU, wer (word *error* rate), and per (position independent word *error* rate) across three language pairs.

method	bleu%	per%	wer%
Geom 1-gram	**14.92**	34.83	46.06
Confidence	**14.74**	35.02	46.11
Random (baseline)	14.11	35.28	46.47

Table 3: Comparison of methods in domain adaptation scenario. The bold numbers show statistically significant improvement with respect to the baseline.

5 Related Work

Despite the promise of active learning for SMT for domain adaptation and low-density/low-resource languages, there has been very little work published on this issue. A Ph.D. proposal by Chris Callison-Burch (Callison-burch, 2003) lays out the promise of AL for SMT and proposes some algorithms. However, the lack of experimental results means that performance and feasibility of those methods cannot be compared to ours. (Mohit and Hwa, 2007) provide a technique to classify phrases as difficult to translate (DTP), and incorporate human translations for these phrases. Their approach is different from AL: they use human translations for DTPs in order to improve translation output in the decoder. There is work on sampling sentence pairs for SMT (Kauchak, 2006; Eck et al., 2005) but the goal

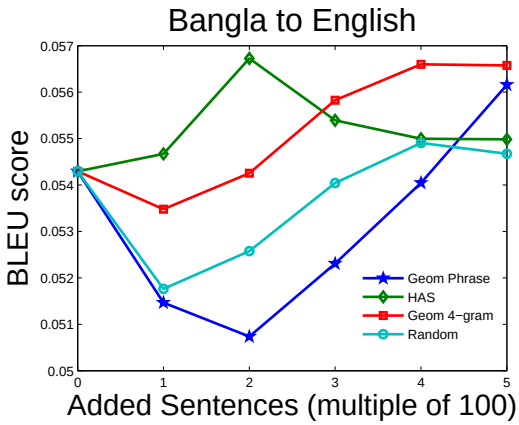


Figure 3: Improving Bangla to English translation performance using active learning.

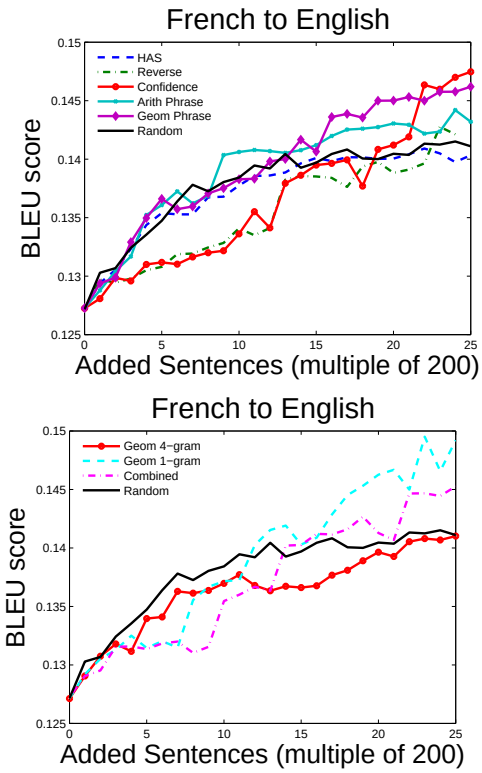


Figure 4: Performance of different sentence selection methods for domain adaptation scenario.

	Fr2En	Ge2En	Sp2En	Ha2En
Avg # of trans	1.30	1.26	1.27	1.30
	1.24	1.25	1.20	1.26
	1.22	1.23	1.19	1.24
	1.22	1.24	1.19	1.24
Avg phrase len	2.85	2.56	2.85	2.85
	3.47	2.74	3.54	3.17
	3.95	3.34	3.94	3.48
	3.58	2.94	3.63	3.36
# of phrases	27,566	29,297	30,750	27,566
	78,026	64,694	93,593	108,787
	79,343	63,191	93,276	115,177
	77,394	65,198	94,597	115,671
# unique events	31,824	33,141	34,937	31,824
	103,124	84,512	125,094	117,214
	86,210	69,357	100,176	127,314
	84,787	72,280	101,636	128,912

Table 4: Average number of english phrases per source language phrase, average length of the source language phrases, number of source language phrases, and number of phrase pairs which has been seen once in the phrase tables across three language pairs (French text taken from Hansard is abbreviated by 'Ha'). From top to bottom in each row, the numbers belong to: before starting AL, and after finishing AL based on 'Geom Phrase', 'Confidence', and 'Random'.

has been to limit the amount of training data in order to reduce the memory footprint of the SMT decoder. To compute this score, (Eck et al., 2005) use n-gram features very different from the n-gram features proposed in this paper. (Kato and Barnard, 2007) implement an AL system for SMT for language pairs with limited resources (En-Xhosa, En-Zulu, En-Setswana and En-Afrikaans), but the experiments are on a very small simulated data set. The only feature used is the confidence score of the SMT system, which we showed in our experiments is not a reliable feature.

6 Conclusions

We provided a novel active learning framework for SMT which utilizes both labeled and unlabeled data. Several sentence selection strategies were proposed and comprehensively compared across three simulated language pairs and a realistic setting of Bangla-English translation with scarce resources. Based on our experiments, we conclude that paying attention to units of translations, i.e. words and candidate phrases in particular, is essential to sentence selection in AL-SMT. Increasing the coverage of the bilingual training data is important but is not the only factor (see Table 4 and Fig. 5). For example, decoder confidence for sentence selection has low coverage (in terms of new words), but performs well in the domain adaptation scenario and performs poorly otherwise. In future work, we plan to explore selection methods based on potential phrases, adaptive sampling using features other than decoder confidence and the use of features from confidence estimation in MT (Ueffing and Ney, 2007).

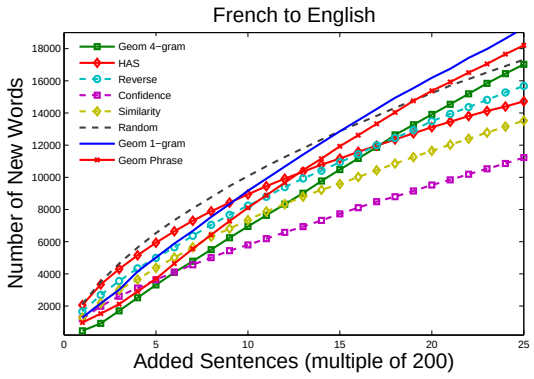


Figure 5: Number of words in domain adaptation scenario.

References

Chris Callison-burch. 2003. Active learning for statistical machine translation. In *PhD Proposal, Edinburgh University*.

David Cohn, Les Atlas, and Richard Ladner. 1994. Improving generalization with active learning. In *Machine Learning Journal*.

Sanjoy Dasgupta and Daniel Hsu. 2008. Hierarchical sampling for active learning. In *proceedings of International Conference on Machine Learning*.

Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Low cost portability for statistical machine translation based in n-gram frequency and tf-idf. In *proceedings of International Workshop on Spoken Language Translation (IWSLT)*.

R.S.M. Kato and E. Barnard. 2007. Statistical translation with scarce resources: a south african case study. *SAIEE Africa Research Journal*, 98(4):136–140, December.

David Kauchak. 2006. Contribution to research on machine translation. In *PhD Thesis, University of California at San Diego*.

Behrang Mohit and Rebecca Hwa. 2007. Localization of difficult-to-translate phrases. In *proceedings of the 2nd ACL Workshop on Statistical Machine Translations*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Roi Reichart, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *proceedings of International Conference on Spoken Language Processing (ICSLP)*.

Marco Turchi, Tijl De Bie, and Nello Cristianini. 2008. Learning performance of a machine translation system: a statistical and computational analysis. In *proceedings of the Third Workshop on Statistical Machine Translation*. Association for Computational Linguistics (ACL).

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

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

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. 2007b. Transductive learning for statistical machine translation. In *proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Semi-Supervised Lexicon Mining from Parenthetical Expressions in Monolingual Web Pages

Xianchao Wu[†] **Naoaki Okazaki**[†] **Jun'ichi Tsujii**[†‡]

[†]Computer Science, Graduate School of Information Science and Technology, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
[‡]School of Computer Science, University of Manchester
National Centre for Text Mining (NaCTeM)
Manchester Interdisciplinary Biocentre, 131 Princess Street, Manchester M1 7DN, UK

{wxc, okazaki, tsujii}@is.s.u-tokyo.ac.jp

Abstract

This paper presents a semi-supervised learning framework for mining Chinese-English lexicons from large amount of Chinese Web pages. The issue is motivated by the observation that many Chinese neologisms are accompanied by their English translations in the form of parenthesis. We classify parenthetical translations into bilingual abbreviations, transliterations, and translations. A frequency-based term recognition approach is applied for extracting bilingual abbreviations. A self-training algorithm is proposed for mining transliteration and translation lexicons. In which, we employ available lexicons in terms of morpheme levels, i.e., phoneme correspondences in transliteration and grapheme (e.g., suffix, stem, and prefix) correspondences in translation. The experimental results verified the effectiveness of our approaches.

1 Introduction

Bilingual lexicons, as lexical or phrasal parallel corpora, are widely used in applications of multilingual language processing, such as statistical machine translation (SMT) and cross-lingual information retrieval. However, it is a time-consuming task for constructing large-scale bilingual lexicons by hand. There are many facts cumber the manual development of bilingual lexicons, such as the continuous emergence of neologisms (e.g., new technical terms, personal names, abbreviations, etc.), the difficulty of keeping up with the neologisms for lexicographers, etc. In order to turn the facts to a better way, one of the simplest strategies is to automatically mine large-scale lexicons from corpora such as the daily updated Web.

Generally, there are two kinds of corpora used for automatic lexicon mining. One is the purely monolingual corpora, wherein frequency-based expectation-maximization (EM, refer to (Dempster et al., 1977)) algorithms and cognate clues play a central role (Koehn and Knight, 2002). Haghighi et al. (2008) presented a generative model based on canonical correlation analysis, in which monolingual features such as the context and orthographic substrings of words were taken into account. The other is multilingual parallel and comparable corpora (e.g., Wikipedia[1]), wherein features such as co-occurrence frequency and context are popularly employed (Cheng et al., 2004; Shao and Ng, 2004; Cao et al., 2007; Lin et al., 2008).

In this paper, we focus on a special type of comparable corpus, *parenthetical translations*. The issue is motivated by the observation that Web pages and technical papers written in Asian languages (e.g., Chinese, Japanese) sometimes annotate named entities or technical terms with their translations in English inside a pair of parentheses. This is considered to be a traditional way to annotate new terms, personal names or other named entities with their English translations expressed in brackets. Formally, a parenthetical translation can be expressed by the following pattern,

$$f_1\ f_2\ ...\ f_J\ (e_1\ e_2\ ...\ e_I). \quad (1)$$

Here, $f_1\ f_2\ ...\ f_J (f_1^J)$, the pre-parenthesis text, denotes the word sequence of some language other than English; and $e_1\ e_2\ ...\ e_I (e_1^I)$, the in-parenthesis text, denotes the word sequence of English. We separate parenthetical translations into three categories:

[1]http://en.wikipedia.org/wiki/Main_Page

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 424–432,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

Type	Examples with translations in *italic*
Abbreviation	对‖全球 气候 观测 系统 (GCOS) *to Global Climate Observing System (GCOS)*
Transliteration	品牌 将 在‖辛普顿- 特尔曼(Shipton-Tilman) *brand will be among Shipton-Tilman (Shipton-Tilman)*
Translation	定时炸弹，‖删除蝇(Cancelbots) *time bomb, Cancelbots (Cancelbots)*
Mixture	在 香港 上课 的 英国‖布拉福特 大学 (Bradford University) *the English Bradford University (Bradford University) that holds lessons in Hongkong*

Table 1: Parenthetical translation categories and examples extracted from Chinese Web pages. Mixture stands for the mixture of translation (*University*) and transliteration (*Bradford*). '‖' denotes the left boundary of f_1^J.

bilingual abbreviation, transliteration, and translation. Table 1 illustrates examples of these categories.

We address several characteristics of parenthetical translations that differ from traditional comparable corpora. The first is that they only appear in monolingual Web pages or documents, and the context information of e_1^I is unknown. Second, frequency and word number of e_1^I are frequently small. This is because parenthetical translations are only used when the authors thought that f_1^J contained some neologism(s) which deserved further explanation in another popular language (e.g., English). Thus, traditional context based approaches are not applicable and frequency based approaches may yield low recall while with high precision. Furthermore, cognate clues such as orthographic features are not applicable between language pairs such as English and Chinese.

Parenthetical translation mining faces the following issues. First, we need to distinguish parenthetical translations from parenthetical expressions, since parenthesis has many functions (e.g., defining abbreviations, elaborations, ellipsis, citations, annotations, etc.) other than translation. Second, the left boundary (denoted as ‖ in Table 1) of the pre-parenthesis text need to be determined to get rid of the unrelated words. Third, we need further distinguish different translation types, such as bilingual abbreviation, the mixture of translation and transliteration, as shown in Table 1.

In order to deal with these problems, supervised (Cao et al., 2007) and unsupervised (Li et al., 2008) methods have been proposed. However, supervised approaches are restricted by the quality and quantity of manually constructed training data, and unsupervised approaches are totally frequency-based without using any semantic clues. In contrast, we propose a semi-supervised framework for mining parenthetical translations. We apply a monolingual abbreviation extraction approach to bilingual abbreviation extraction. We construct an English-syllable to Chinese-pinyin transliteration model which is self-trained using phonemic similarity measurements. We further employ our cascaded translation model (Wu et al., 2008) which is self-trained based on morpheme-level translation similarity.

This paper is organized as follows. We briefly review the related work in the next section. Our system framework and self-training algorithm is described in Section 3. Bilingual abbreviation extraction, self-trained transliteration models and cascaded translation models are described in Section 4, 5, and 6, respectively. In Section 7, we evaluate our mined lexicons by Wikipedia. We conclude in Section 8 finally.

2 Related Work

Numerous researchers have proposed a variety of automatic approaches to mine lexicons from the Web pages or other large-scale corpora. Shao and Ng (2004) presented a method to mine new translations from Chinese and English news documents of the same period from different news agencies, combining both transliteration and context information. Kuo et al. (2006) used active learning and unsupervised learning for mining transliteration lexicon from the Web pages, in which an EM process was used for estimating the phonetic similarities between English syllables and Chinese characters.

Cao et al. (2007) split parenthetical translation mining task into two parts, transliteration detection and translation detection. They employed a transliteration lexicon for constructing a grapheme-based transliteration model and annotated boundaries manually to train a classifier. Lin et al. (2008) applied a frequency-based word alignment approach, Competitive Link (Melanmed, 2000), to determine the outer boundary (Section 7).

On the other hand, there have been many semi-supervised approaches in numerous applications

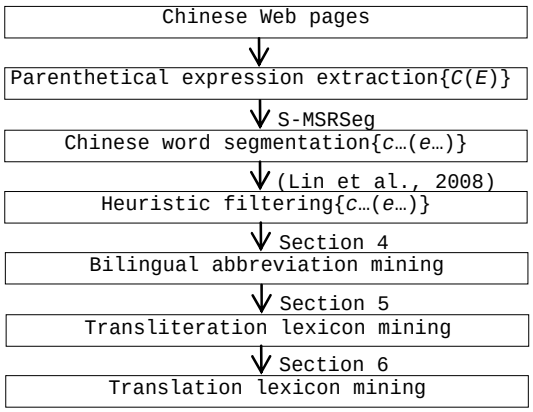


Figure 1: The system framework of mining lexicons from Chinese Web pages.

(Zhu, 2007), such as self-training in word sense disambiguation (Yarowsky, 2005) and parsing (McClosky et al., 2008). In this paper, we apply self-training to a new topic, lexicon mining.

3 System Framework and Self-Training Algorithm

Figure 1 illustrates our system framework for mining lexicons from Chinese Web pages. First, parenthetical expressions matching Pattern 1 are extracted. Then, pre-parenthetical Chinese sequences are segmented into word sequences by S-MSRSeg[2] (Gao et al., 2006). The initial parenthetical translation corpus is constructed by applying the heuristic rules defined in (Lin et al., 2008)[3]. Based on this corpus, we mine three lexicons step by step, a bilingual abbreviation lexicon, a transliteration lexicon, and a translation lexicon. The abbreviation candidates are extracted firstly by using a heuristic rule (Section 4.1). Then, the transliteration candidates are selected by employing a transliteration model (Section 5.1). Specially, $f_1^J(e_1^I)$ is taken as a transliteration candidate only if a word e_i in e_1^I can be transliterated. In addition, a transliteration candidate will also be considered as a translation candidate if not all e_i can be transliterated (refer to the mixture example in Table1). Finally, after abbreviation filtering and transliteration filtering, the remaining candidates are used for translation lexicon mining.

[2]http://research.microsoft.com/research/downloads/details/7a2bb7ee-35e6-40d7-a3f1-0b743a56b424/details.aspx

[3]e.g., f_1^J is predominantly in Chinese and e_1^I is predominantly in English

Algorithm 1 self-training algorithm

Require: L, $U = \{f_1^J(e_1^I)\}$, T, $\mathcal{M}$ $\quad \triangleright L$, (labeled) training set; U, (unlabeled) candidate set; T, test set; $\mathcal{M}$, the transliteration or translation model.

1: $Lexicon = \{\}$ $\quad \triangleright$ new mined lexicon
2: **repeat**
3: $\quad N = \{\} \triangleright$ new mined lexicon during one iteration
4: $\quad$ train $\mathcal{M}$ on L
5: $\quad$ evaluate $\mathcal{M}$ on T
6: $\quad$ **for** $f_1^J(e_1^I) \in U$ **do**
7: $\quad\quad topN = \{C' | \text{decode } e_1^I \text{ by } \mathcal{M}\}$
8: $\quad\quad N = N \cup \{(c, e_1^I) | c \in f_1^J \wedge \exists C' \in topN \; s.t. \; similarity\{c, C'\} \geq \theta\}$
9: $\quad$ **end for**
10: $\quad U = U - N$
11: $\quad L = unified(L \cup N)$
12: $\quad Lexicon = unified(Lexicon \cup N)$
13: **until** $|N| \leq \epsilon$
14: **return** $Lexicon$ $\quad \triangleright$ the output

Algorithm 1 addresses the self-training algorithm for lexicon mining. The main part is a loop from Line 2 to Line 13. A given seed lexicon is taken as labeled data and is split into training and testing sets (L and T). U=$\{f_1^J(e_1^I)\}$, stands for the (unlabeled) parenthetical expression set. Initially, a translation/transliteration model ($\mathcal{M}$) is trained on L and evaluated on T (Line 4 and 5). Then, the English phrase e_1^I of each unlabeled entry is decoded by $\mathcal{M}$, and the top-N outputs are stored in set $topN$ (Line 7$\sim$8). A similarity function on c (a word substring of f_1^J) and a top-N output C' is employed to make the decision of classification: the pair (c, e_1^I) will be selected as a new entry if the similarity between c and C' is no smaller than a threshold value θ (Line 8). After processing each entry in U, the new mined lexicon N is deleted from U and unified with the current training set L as the new training set (Line 10 and 11). Also, N is added to the final lexicon (Line 12). When $|N|$ is lower than a threshold, the loop stops. Finally, the algorithm returns the mined lexicon.

One of the open problems in Algorithm 1 is how to append new mined entries into the existing seed lexicon, considering they have different distributions. One way is to design and estimate a weight function on the frequency of new mined entries. For simplicity, we use a deficient strategy that takes the weights of all new mined entries to be one.

4 Bilingual Abbreviation Extraction

4.1 Methodology

The method that we use for extracting a bilingual abbreviation lexicon from parenthetical expressions is inspired by (Okzaki and Ananiadou, 2006). They used a term recognition approach to build a monolingual abbreviation dictionary from the Medical Literature Analysis and Retrieval System Online (MEDLINE) abstracts, wherein acronym definitions (e.g., *ADM* is short for *adriamycin*, *adrenomedullin*, etc.) are abundant. They reported 99% precision and 82-95% recall. Through locating a textual fragment with an acronym and its expanded form in pattern

$$\text{long form (short form)}, \tag{2}$$

they defined a heuristic formula to compute the long-form likelihood LH(c) for a candidate c:

$$\text{LH}(c) = \text{freq}(c) - \sum_{t \in T_c} \text{freq}(t) \times \frac{\text{freq}(t)}{\sum_{t \in T_c} \text{freq}(t)}. \tag{3}$$

Here, c is a long-form candidate; freq(c) denotes the frequency of co-occurrence of c with a short-form; and T_c is a set of nested long-form candidates, each of which consists of a preceding word followed by the candidate c. Obviously, for $t \in T_c$, Equation 3 can be explained as:

$$\text{LH}(c) = \text{freq}(c) - \mathbb{E}[\text{freq}(t)]. \tag{4}$$

In this paper, we apply their method on the task of bilingual abbreviation lexicon extraction. Now, the long-form is a Chinese word sequence and the short-form is an English acronym. We filter the parenthetical expressions in the Web pages with several heuristic rules to meet the form of pattern 2 and to save the computing time:

- the short-form (e_1^I) should contain only one English word ($I = 1$), and all letters in which should be capital;
- similar with (Lin et al., 2008), the pre-parenthesis text is trimmed with: $|c| \geq 10 \times |e_1^I| + 6$ when $|e_1^I| \leq 6$, and $|c| \geq 2 \times |e_1^I| + 6$, otherwise. $|c|$ and $|e_1^I|$ are measured in bytes. We further trim the remaining pre-parenthesis text by punctuations other than hyphens and dots, i.e., the right most punctuation and its left subsequence are discarded.

No.	Chinese long-form candidates	LH	T/F
1	肿瘤 相关 抗原 *Tumor-Associated Antigen*	172.5	T
2	硫 代 乙酰 胺 *thioacetamide*	79.9	T
3	胺 *amine*	33.8	F
4	抗原 *antigen*	24.5	F
5	相关 抗原 *associated antigen*	21.2	F
6	的 肿瘤 相关 抗原 *'s Tumor-Associated Antigen*	16.5	F
7	总 氨基酸 *total amino acid*	16.2	T

Table 2: Top-7 Chinese long-form candidates for the English acronym *TAA*, according to the LH score.

4.2 Experiment

We used SogouT Internet Corpus Version 2.0[4], which contains about 13 billion original Web pages (mainly Chinese) in the form of 252 gigabyte .txt files. In addition, we used 55 gigabyte (.txt format) Peking University Chinese Paper Corpus. We constructed a partially parallel corpus in the form of Pattern 1 from the union of the two corpora using the heuristic rules defined in (Lin et al., 2008). We gained a partially parallel corpus which contains 12,444,264 entries.

We extracted 107,856 distinct English acronyms. Limiting LH score ≥ 1.0 in Equation 3, we gained 2,020,012 Chinese long-form candidates for the 107,856 English acronyms. Table 2 illustrates the top-7 Chinese long-form candidates of the English acronym *TAA*. Three candidates are correct (T) long-forms while the other 4 are wrong (F). Wrong candidates from No. 3 to 5 are all subsequences of the correct candidate No. 1. No. 6 includes No. 1 while with a Chinese functional word *de* in the left most side. These error types can be easily tackled with some filtering patterns, such as 'remove the left most functional word in the long-form candidates', 'only keep the relatively longer candidates with larger LH score', etc.

Since there does not yet exists a common evaluation data set for the bilingual abbreviation lexicon, we manually evaluated a small sample of it.

[4]http://www.sogou.com/labs/dl/t.html

Of the 107,856 English acronyms, we randomly selected 200 English acronyms and their top-1 Chinese long-form candidates for manually evaluating. We found, 92 candidates were correct including 3 transliteration examples. Of the 108 wrong candidates, 96 candidates included the correct long-form with some redundant words on the left side (i.e., $c = (\text{word})^+$ correct long-form), the other 12 candidates missed some words of the correct long-form or had some redundant words right before the left parenthesis (i.e., $c = (\text{word})^*$ correct long-form $(\text{word})^+$ or $c = (\text{word})^*$ subsequence of correct long-form $\text{word})^*$). We classified the redundant word right before the correct long-form of each of the 96 candidates, *de* occupied 32, noun occupied 7, verb occupied 18, prepositions and conjunctions occupied the remaining ones.

In total, the abbreviation translation accuracy is 44.5%. We improved the accuracy to 60.5% with an additional *de* filtering pattern. According to former mentioned error analysis, the accuracy may further be improved if a Chinese part-of-speech tagger is employed and the non-nominal words in the long-form are removed beforehand.

5 Self-Training for Transliteration Models

In this section, we first describe and compare three transliteration models. Then, we select and train the best model following Algorithm 1 for lexicon mining. We investigate two things, the scalability of the self-trained model given different amount of initial training data, and the performance of several strategies for selecting new training samples.

5.1 Model description

We construct and compare three forward transliteration models, a phoneme-based model (English phonemes to Chinese pinyins), a grapheme-based model (English syllables to Chinese characters) and a hybrid model (English syllables to Chinese pinyins). Similar models have been compared in (Oh et al., 2006) for English-to-Korean and English-to-Japanese transliteration. All the three models are phrase-based, i.e., adjacent phonemes or graphemes are allowable to form phrase-level transliteration units. Building the correspondences on phrase level can effectively tackle the missing or redundant phoneme/grapheme problem during transliteration. For example, when *Aamodt* is transliterated into *a mō tè*[5], *a* and *d* are missing. The problem can be easily solved when taking *Aa* and *dt* as single units for transliterating.

Making use of Moses (Koehn et al., 2007), a phrase-based SMT system, Matthews (2007) has shown that the performance was comparable to recent state-of-the-art work (Jiang et al., 2007) in English-to-Chinese personal name transliteration. Matthews (2007) took transliteration as translation at the surface level. Inspired by his idea, we also implemented our transliteration models employing Moses. The main difference is that, while Matthews (2007) tokenized the English names into individual letters before training in Moses, we split them into syllables using the heuristic rules described in (Jiang et al., 2007), such that one syllable only contains one vowel letter or a combination of a consonant and a vowel letter.

English syllable sequences are used in the grapheme-based and hybrid models. In the phoneme-based model, we transfer English names into phonemes and Chinese characters into Pinyins in virtue of the CMU pronunciation dictionary[6] and the LDC Chinese character-to-pinyin list[7].

In the mass, the grapheme-based model is the most robust model, since no additional resources are needed. However, it suffers from the Chinese homophonic character problem. For instance, pinyin ai corresponds to numerous Chinese characters which are applicable to personal names. The phoneme-based model is the most suitable model that reflects the essence of transliteration, while restricted by additional grapheme to phoneme dictionaries. In order to eliminate the confusion of Chinese homophonic characters and alleviate the dependency on additional resources, we implement a hybrid model that accepts English syllables and Chinese pinyins as formats of the training data. This model is called hybrid, since English syllables are graphemes and Chinese pinyins are phonemes.

[5]The tones of Chinese pinyins are ignored in our transliteration models for simplicity.

[6]http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[7]http://projects.ldc.upenn.edu/Chinese/docs/char2pinyin.txt

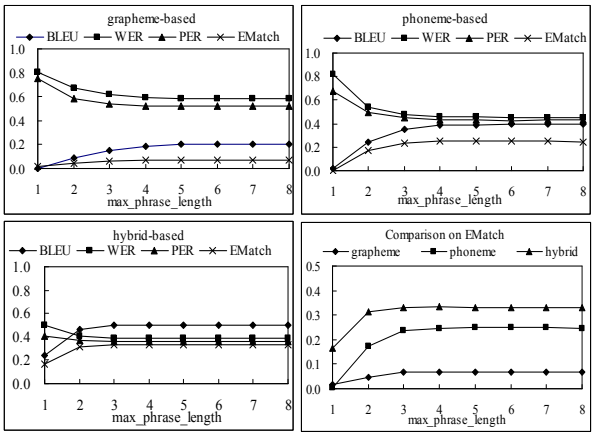


Figure 2: The performances of the transliteration models and their comparison on EMatch.

%	0t	1t	2t	3t	4t	5t	Strategy
5	.3879	.3937	.3971	.3958	.3972	.3971	top1 *em*
		.3911	.3979	.3954	.3974	.3965	top1 *am*
		.4062	.4182	.4208	**.4218**	.4201	top5 *em*
		.3987	.4177	.4190	.4192	.4189	top5 *am*
10	.4092	.4282	.4258	.4202	.4203	.4205	top1 *em*
		.4121	.4190	.4180	.4174	.4200	top1 *am*
		.4305	.4386	.4399	**.4438**	.4403	top5 *em*
		.4289	.4263	.4292	.4291	.4288	top5 *am*
20	.4561	.4538	.4562	.4550	.4543	.4551	top1 *em*
		.4532	.4578	.4544	.4545	.4541	top1 *am*
		.4624	**.4762**	.4754	.4748	.4746	top5 *em*
		.4605	.4677	.4677	.4674	.4679	top5 *am*
40	.4779	.4791	.4793	.4799	.4794	.4808	top1 *em*
		.4774	.4794	.4779	.4789	.4784	top1 *am*
		.4808	**.4811**	.4791	.4795	.4790	top5 *em*
		.4775	.4778	.4781	.4785	.4779	top5 *am*
60	.5032	.4939	.5004	.5012	.5012	.5016	top1 *em*
		.4919	.4988	.4990	.4994	.4990	top1 *am*
		.5013	.5063	.5059	**.5066**	.5065	top5 *em*
		.4919	.4960	.4970	.4977	.4962	top5 *am*
80	.5038	.4984	.4984	.5004	.5006	.4995	top1 *em*
		.4916	.4916	.4914	.4915	.4916	top1 *am*
		.5039	.5037	.5053	**.5054**	.5042	top5 *em*
		.4950	.5028	.5027	.5032	.5032	top5 *am*
100	.5045	.5077	.5053	.5067	.5063	.5066	top1 *em*
		.5045	.5054	.5046	.5050	.5055	top1 *am*
		.5108	.5102	.5111	.5108	**.5115**	top5 *em*
		.5105	.5106	.5100	.5094	.5109	top5 *am*

Table 3: The BLEU score of self-trained h4 transliteration models under four selection strategies. nt (n=1..5) stands for the n-th iteration.

5.2 Experimental model selection

Similar to (Jiang et al., 2007), the transliteration models were trained and tested on the LDC Chinese-English Named Entity Lists Version 1.0[8]. The original list contains 572,213 English people names with Chinese transliterations. We extracted 74,725 entries in which the English names also appeared in the CMU pronunciation dictionary. We randomly selected 3,736 entries as an open testing set and the remaining entries as a training set[9]. The results were evaluated using the character/pinyin-based 4-gram BLEU score (Papineni et al., 2002), word error rate (WER), position independent word error rate (PER), and exact match (EMatch).

Figure 2 reports the performances of the three models and the comparison based on EMatch. From the results, we can easily draw the conclusion that the hybrid model performs the best under the maximal phrase length (mpl, the maximal phrase length allowed in Moses) from 1 to 8. The performances of the models converge at or right after mpl = 4. The pinyin-based WER of the hybrid model is 39.13%, comparable to the pinyin error rate 39.6%, reported in (Jiang et al., 2007)[10]. Thus, our further self-training experiments are pursued on the hybrid model taking mpl to be 4 (short for h4, hereafter).

5.3 Experiments on the self-trained hybrid model

As former mentioned, we investigate the scalability of the self-trained h4 model by respectively using 5, 10, 20, 40, 60, 80, and 100 percent of initial training data, and the performances of using exact matching (*em*) or approximate matching (*am*, line 8 in Algorithm 1) on the top-1 and top-5 outputs (line 7 in Algorithm 1) for selecting new training samples. We used edit distance (ed) to measure the *em* and *am* similarities:

$$ed(c, C') = 0 \text{ or } < \text{syllable\_number}(C')/2. \quad (5)$$

When applying Algorithm 1 for transliteration lexicon mining, we decode each word in e_1^I respectively. The algorithm terminated in five iterations when we set the terminal threshold ϵ (Line 13 in Algorithm 1) to be 100.

For simplicity, Table 3 only illustrates the BLEU score of h4 models under four selection strategies. From this table, we can draw the following conclusions. First, with fewer initial training data, the improvement is better. The best relative improvements

[8]Linguistic Data Consortium catalog number: LDC2005T34 (former catalog number: LDC2003E01)

[9]Jiang et al. (2007) selected 25,718 personal name pairs from LDC2003E01 as the experiment data: 200 as development set, 200 as test set, and the remaining entries as training set.

[10]It should be notified that we achieved this result by using larger training set (70,989 vs. 25,718) and larger test set (3,736 vs. 200) comparing with (Jiang et al., 2007), and we did not use additional Web resources as Jiang et al. (2007) did.

are 8.74%, 8.46%, 4.41%, 0.67%, 0.68%, 0.32%, and 1.39%, respectively. Second, using top-5 and *em* for new training data selection performs the best among the four strategies. Compared under each iteration, using top-5 is better than using top-1; *em* is better than *am*; and top-5 with *am* is a little better than top-1 with *em*. We mined 39,424, 42,466, 46,116, 47,057, 49,551, 49,622, and 50,313 distinct entries under the six types of initial data with top-5 plus *em* strategy. The 50,313 entries are taken as the final transliteration lexicon for further comparison.

6 Self-Training for a Cascaded Translation Model

We classify the parenthetical translation candidates by employing a translation model. In contrast to (Lin et al., 2008), wherein the lengthes of prefixes and suffixes of English words were assumed to be three bytes, we segment words into morphemes (sequences of prefixes, stems, and suffixes) by Morfessor 0.9.2[11], an unsupervised language-independent morphological analyzer (Creutz and Lagus, 2007). We use the morpheme-level translation similarity explicitly in our cascaded translation model (Wu et al., 2008), which makes use of morpheme, word, and phrase level translation units. We train Moses to gain a phrase-level translation table. To gain a morpheme-level translation table, we run GIZA++ (Och and Ney, 2003) on both directions between English morphemes and Chinese characters, and take the intersection of *Viterbi* alignments. The English-to-Chinese translation probabilities computed by GIZA++ are attached to each morpheme-character element in the intersection set.

6.1 Experiment

The Wanfang Chinese-English technical term dictionary[12], which contains 525,259 entries in total, was used for training and testing. 10,000 entries were randomly selected as the test set and the remaining as the training set. Again, we investigated the scalability of the self-trained cascaded translation model by respectively using 20, 40, 60, 80, and 100 percent of initial training data. An aggressive similarity measurement was used for selecting new training samples:

$$\text{first\_char}(c) = \text{first\_char}(C') \ \wedge \ min\{ed(c, C')\}. \quad (6)$$

Here, we judge if the first characters of c and C' are similar or not. c was gained by deleting zero or more characters from the left side of f_1^J. When more than one c satisfied this condition, the c that had the smallest edit distance with C' was selected. When applying Algorithm 1 for translation lexicon mining, we took e_1^I as one input for decoding instead of decoding each word respectively. Only the top-1 output (C') was used for comparing. The algorithm stopped in five iterations when we set the terminal threshold ϵ to be 2000.

%	0t	1t	2t	3t	4t	5t
20	**.1406**	.1196	.1243	.1239	.1176	.1179
40	.1091	.1224	.1386	.1345	**.1479**	.1466
60	.1630	.1624	.1429	**.1714**	.1309	.1398
80	**.1944**	.1783	.1886	.1870	.1884	.1873
100	.1810	.1814	.1539	**.1981**	.1542	.1944

Table 4: The BLEU score of self-trained cascaded translation model under five initial training sets.

For simplicity, Table 4 only illustrates the BLEU score of the cascaded translation model under five initial training sets. For the reason that there are finite phonemes in English and Chinese while the semantic correspondences between the two languages tend to be infinite, Table 4 is harder to be analyzed than Table 3. When initially using 40%, 60%, and 100% training data for self-training, the results tend to be better at some iterations. We gain 35.6%, 5.2%, and 9.4% relative improvements, respectively. However, the results tend to be worse when 20% and 80% training data were used initially, with 11.6% and 3.0% minimal relative loss. The best BLEU scores tend to be better when more initial training data are available. We mined 1,038,617, 1,025,606, 1,048,761, 1,056,311, and 1,060,936 distinct entries under the five types of initial training data. The 1,060,936 entries are taken as the final translation lexicon for further comparison.

7 Wikipedia Evaluation

We have mined three kinds of lexicons till now, an abbreviation lexicon containing 107,856 dis-

[11] http://www.cis.hut.fi/projects/morpho/

[12] http://www.wanfangdata.com.cn/Search/ResourceBrowse.aspx

	En. to Ch.		Ch. to En.	
	Cov	EMatch	Cov	EMatch
Our Lexicon	22.8%	5.2%	23.2%	5.5%
Unsupervised	23.5%	5.4%	24.0%	5.4%

Table 5: The results of our lexicon and an unsupervised-mined lexicon (Lin et al., 2008) evaluated under Wikipedia title dictionary. Cov is short for coverage.

similar English acronyms with 2,020,012 Chinese long-form candidates; a transliteration lexicon with 50,313 distinct entries; and a translation lexicon with 1,060,936 distinct entries. The three lexicons are combined together as our final lexicon.

Similar with (Lin et al., 2008), we compare our final mined lexicon with a dictionary extracted from Wikipedia, the biggest multilingual free-content encyclopedia on the Web. We extracted the titles of Chinese and English Wikipedia articles[13] that are linked to each other. Since most titles contain less than five words, we take a linked title pair as a translation entry without considering the word alignment relation between the words inside the titles. The result lexicon contains 105,320 translation pairs between 103,823 Chinese titles and 103,227 English titles. Obviously, only a small percentage of titles have more than one translation. Whenever there is more than one translation, we take the candidate entry as correct if and only if it matches one of the translations.

Moreover, we compare our semi-supervised approach with an unsupervised approach (Lin et al., 2008). Lin et al. (2008) took $\varphi^2(f_j, e_i)$ score [14](Gale and Church, 1991) with threshold 0.001 as the word alignment probability in a word alignment algorithm, Competitive Link. Competitive Link tries to align an unlinked e_i with an unlinked f_j by the condition that $\varphi^2(f_j, e_i)$ is the biggest. Lin et al. (2008) relaxed the unlinked constraints to allow consecutive sequence of words on one side to be linked to the same word on the other side[15]. The left boundary inside f_1^J is determined when each e_i in e_1^I is aligned. After applying the modified Competitive Link on the partially parallel corpus which includes 12,444,264 entries (Section 4.2), we obtained 2,628,366 distinct pairs.

Table 5 shows the results of the two lexicons evaluated under Wikipedia title dictionary. The coverage is measured by the percentage of titles which appears in the mined lexicon. We then check whether the translation in the mined lexicon is an exact match of one of the translations in the Wikipedia lexicon. Through comparing the results, our mined lexicon is comparable with the lexicon mined in an unsupervised way. Since the selection is based on phonemic and semantic clues instead of frequency, a parenthetical translation candidate will not be selected if the in-parenthetical English text is failed to be transliterated or translated. This is one reason that explains why we earned a little lower coverage. Another reason comes from the low coverage rate of seed lexicons used for self-training, only 8.65% English words in the partially parallel corpus are covered by the Wanfang dictionary.

8 Conclusion

We have proposed a semi-supervised learning framework for mining bilingual lexicons from parenthetical expressions in monolingual Web pages. We classified the parenthesis expressions into three categories: abbreviation, transliteration, and translation. A set of heuristic rules, a self-trained hybrid transliteration model, and a self-trained cascaded translation model were proposed for each category, respectively.

We investigated the scalability of the self-trained transliteration and translation models by training them with different amount of data. The results shew the stability (transliteration) and feasibility (translation) of our proposals. Through employing the parallel Wikipedia article titles as a gold standard lexicon, we gained the comparable results comparing our semi-supervised framework with our implementation of Lin et al. (2008)'s unsupervised mining approach.

[13]English and Chinese Wikipedia pages due to 2008.09.23 are used here.

[14]$\varphi^2(f_j, e_i) = \frac{(ad-bc)^2}{(a+b)(a+c)(b+d)(c+d)}$, where a is the number of $f_1^J(e_1^I)$ containing both e_i and f_j; $(a+b)$ is the number of $f_1^J(e_1^I)$ containing e_i; $(a+c)$ is the number of $f_1^J(e_1^I)$ containing f_j; and d is the number of $f_1^J(e_1^I)$ containing neither e_i nor f_j.

[15]Instead of requiring both e_i and f_j to have no previous linkages, they only require that at least one of them be unlinked and that (suppose e_i is unlinked and f_j is linked to e_k) none of the words between e_i and e_k be linked to any word other than f_j.

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan) and Japanese/Chinese Machine Translation Project in Special Coordination Funds for Promoting Science and Technology (MEXT, Japan). We thank the anonymous reviewers for their constructive comments.

References

Cao, Guihong, Jianfeng Gao, and Jian-Yun Nie. 2007. A system to Mine Large-Scale Bilingual Dictionaries from Monolingual Web Pages. In *MT Summit XI*. pages 57–64, Copenhagen, Denmark.

Cheng, Pu-Jen, Yi-Cheng Pan, Wen-Hsiang Lu, and Lee-Feng Chien. 2004. Creating Multilingual Translation Lexicons with Regional Variations Using Web Corpora. In *ACL 2004*, pages 534–541, Barcelona, Spain.

Creutz, Mathias and Krista Lagus. 2007. Unsupervised Models for Morpheme Segmentation and Morphology Learning. *ACM Transactions on Speech and Language Processing*, 4(1):Article 3.

Dempster, A. P., N. M. Laird and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39:1–38.

Gale, W. and K. Church. 1991. Identifying word correspondence in parallel text. In *DARPA NLP Workshop*.

Gao, Jianfeng, Mu Li, Andi Wu, and Chang-Ning Huang. 2006. Chinese Word Segmentation and Named Entity Recognition: A Pragmatic Approach. *Computational Linguistics*, 31(4):531–574.

Haghighi, Aria, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein 2008. Learning Bilingual Lexicons from Monolingual Corpora. In *ACL-08:HLT*. pages 771–779, Columbus, Ohio.

Jiang, Long, Ming Zhou, Lee-Feng Chien, and Cheng Niu. 2007. Named Entity Translation with Web Mining and Transliteration. In *IJCAI 2007*. pages 1629–1634, Hyderabad, India.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *ACL 2007 Poster Session*, pages 177–180.

Koehn, Philipp and Kevin Knight. 2002. Learning a translation lexicon from monolingual corpora. In *SIGLEX 2002*, pages 9–16.

Kuo, Jin-Shea, Haizhou Li, and Ying-Kuei Yang. 2006. Learning Transliteration Lexicons from the Web. In *COLING-ACL 2006*. pages 1129–1136.

Lin, Dekang, Shaojun Zhao, Benjamin Van Durme, and Marius Paşca. 2008. Mining Parenthetical Translations from the Web by Word Alignment. In *ACL-08:HLT*, pages 994–1002, Columbus, Ohio.

Matthews, David. 2007. Machine Transliteration of Proper Names. A Thesis of Master. University of Edinburgh.

McClosky, David, Eugene Charniak, and Mark Johnson 2008. When is Self-Training Effective for Parsing? In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 561–568, manchester, UK.

Melamed, I. Dan. 2000. Models of Translational Equivalence among Words. *Computational Linguistics*, 26(2):221–249.

Och, Franz Josef and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Oh, Jong-Hoon, Key-Sun Choi, and Hitoshi Isahara. 2006. A Comparison of Different Machine Transliteration Models. *Journal of Artifical Intelligence Research*, 27:119–151.

Okazaki, Naoaki and Sophia Ananiadou. 2006. Building an Abbreviation Dictionary Using a Term Recognition Approach. *Bioinformatics*, 22(22):3089–3095.

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)*. pages 311–318, Philadelphia.

Shao, Li and Hwee Tou Ng. 2004. Mining New Word Translations from Comparable Corpora. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 618–624, Geneva, Switzerland.

Wu, Xianchao, Naoaki Okazaki, Takashi Tsunakawa, and Jun'ichi Tsujii. 2008. Improving English-to-Chinese Translation for Technical Terms Using Morphological Information. In *Proceedings of the 8th Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 202–211, Waikiki, Hawai'i.

Yarowsky, David. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts.

Zhu, Xiaojin. 2007. *Semi-Supervised Learning Literature Survery*. University of Wisconsin - Madison.

Hierarchical Phrase-Based Translation with Weighted Finite State Transducers

Gonzalo Iglesias[⋆] **Adrià de Gispert**[‡] **Eduardo R. Banga**[⋆] **William Byrne**[‡]
[⋆] University of Vigo. Dept. of Signal Processing and Communications. Vigo, Spain
{giglesia,erbanga}@gts.tsc.uvigo.es
[‡] University of Cambridge. Dept. of Engineering. CB2 1PZ Cambridge, U.K.
{ad465,wjb31}@eng.cam.ac.uk

Abstract

This paper describes a lattice-based decoder for hierarchical phrase-based translation. The decoder is implemented with standard WFST operations as an alternative to the well-known cube pruning procedure. We find that the use of WFSTs rather than k-best lists requires less pruning in translation search, resulting in fewer search errors, direct generation of translation lattices in the target language, better parameter optimization, and improved translation performance when rescoring with long-span language models and MBR decoding. We report translation experiments for the Arabic-to-English and Chinese-to-English NIST translation tasks and contrast the WFST-based hierarchical decoder with hierarchical translation under cube pruning.

1 Introduction

Hierarchical phrase-based translation generates translation hypotheses via the application of hierarchical rules in CYK parsing (Chiang, 2005). *Cube pruning* is used to apply language models at each cell of the CYK grid as part of the search for a k-best list of translation candidates (Chiang, 2005; Chiang, 2007). While this approach is very effective and has been shown to produce very good quality translation, the reliance on k-best lists is a limitation. We take an alternative approach and describe a lattice-based hierarchical decoder implemented with Weighted Finite State Transducers (WFSTs). In every CYK cell we build a single, minimal word lattice containing all possible translations of the source sentence span covered by that cell. When derivations contain non-terminals, we use pointers to lower-level lattices for memory efficiency. The pointers are only expanded to the actual translations if pruning is required during search; expansion is otherwise only carried out at the upper-most cell, after the full CYK grid has been traversed.

We describe how this decoder can be easily implemented with WFSTs. For this we employ the OpenFST libraries (Allauzen et al., 2007). Using standard FST operations such as composition, epsilon removal, determinization, minimization and shortest-path, we find this search procedure to be simpler to implement than cube pruning. The main modeling advantages are a significant reduction in search errors, a simpler implementation, direct generation of target language word lattices, and better integration with other statistical MT procedures. We report translation results in Arabic-to-English and Chinese-to-English translation and contrast the performance of lattice-based and cube pruning hierarchical decoding.

1.1 Related Work

Hierarchical phrase-based translation has emerged as one of the dominant current approaches to statistical machine translation. Hiero translation systems incorporate many of the strengths of phrase-based translation systems, such as feature-based translation and strong target language models, while also allowing flexible translation and movement based on hierarchical rules extracted from aligned parallel text. We summarize some extensions to the basic approach to put our work in context.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 433–441,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

Hiero Search Refinements Huang and Chiang (2007) offer several refinements to cube pruning to improve translation speed. Venugopal et al. (2007) introduce a Hiero variant with relaxed constraints for hypothesis recombination during parsing; speed and results are comparable to those of cube pruning, as described by Chiang (2007). Li and Khudanpur (2008) report significant improvements in translation speed by taking unseen n-grams into account within cube pruning to minimize language model requests. Dyer et al. (2008) extend the translation of source sentences to translation of input lattices following Chappelier et al. (1999).

Extensions to Hiero Several authors describe extensions to Hiero, to incorporate additional syntactic information (Zollmann and Venugopal, 2006; Zhang and Gildea, 2006; Shen et al., 2008; Marton and Resnik, 2008), or to combine it with discriminative latent models (Blunsom et al., 2008).

Analysis and Contrastive Experiments Zollman et al. (2008) compare phrase-based, hierarchical and syntax-augmented decoders for translation of Arabic, Chinese, and Urdu into English. Lopez (2008) explores whether lexical reordering or the phrase discontiguity inherent in hierarchical rules explains improvements over phrase-based systems. Hierarchical translation has also been used to great effect in combination with other translation architectures, e.g. (Sim et al., 2007; Rosti et al., 2007).

WFSTs for Translation There is extensive work in using Weighted Finite State Transducer for machine translation (Bangalore and Riccardi, 2001; Casacuberta, 2001; Kumar and Byrne, 2005; Mathias and Byrne, 2006; Graehl et al., 2008).

To our knowledge, this paper presents the first description of hierarchical phrase-based translation in terms of lattices rather than k-best lists. The next section describes hierarchical phrase-based translation with WFSTs, including the lattice construction over the CYK grid and pruning strategies. Section 3 reports translation experiments for Arabic-to-English and Chinese-to-English, and Section 4 concludes.

2 Hierarchical Translation with WFSTs

The translation system is based on a variant of the CYK algorithm closely related to CYK+ (Chappelier and Rajman, 1998). Parsing follows the description of Chiang (2005; 2007), maintaining backpointers and employing hypothesis recombination without pruning. The underlying model is a synchronous context-free grammar consisting of a set $\mathbf{R} = \{R^r\}$ of rules $R^r : N \rightarrow \langle \gamma^r, \alpha^r \rangle$ / p^r, with 'glue' rules, $S \rightarrow \langle X, X \rangle$ and $S \rightarrow \langle S\ X, S\ X \rangle$. If a rule has probability p^r, it is transformed to a cost c^r; here we use the tropical semiring, so $c^r = -\log p^r$. N denotes a non-terminal; in this paper, N can be either S, X, or V (see section 3.2). $\mathbf{T}$ denotes the terminals (words), and the grammar builds parses based on strings $\gamma, \alpha \in \{\{S, X, V\} \cup \mathbf{T}\}^+$. Each cell in the CYK grid is specified by a non-terminal symbol and position in the CYK grid: (N, x, y), which spans s_x^{x+y-1} on the source sentence.

In effect, the source language sentence is parsed using a context-free grammar with rules $N \rightarrow \gamma$. The generation of translations is a second step that follows parsing. For this second step, we describe a method to construct word lattices with all possible translations that can be produced by the hierarchical rules. Construction proceeds by traversing the CYK grid along the backpointers established in parsing. In each cell (N, x, y) in the CYK grid, we build a target language word lattice $\mathcal{L}(N, x, y)$. This lattice contains every translation of s_x^{x+y-1} from every derivation headed by N. These lattices also contain the translation scores on their arc weights.

The ultimate objective is the word lattice $\mathcal{L}(S, 1, J)$ which corresponds to all the analyses that cover the source sentence s_1^J. Once this is built, we can apply a target language model to $\mathcal{L}(S, 1, J)$ to obtain the final target language translation lattice (Allauzen et al., 2003).

We use the approach of Mohri (2002) in applying WFSTs to statistical NLP. This fits well with the use of the OpenFST toolkit (Allauzen et al., 2007) to implement our decoder.

2.1 Lattice Construction Over the CYK Grid

In each cell (N, x, y), the set of rule indices used by the parser is denoted $R(N, x, y)$, *i.e.* for $r \in R(N, x, y)$, $N \rightarrow \langle \gamma^r, \alpha^r \rangle$ was used in at least one derivation involving that cell.

For each rule $R^r, r \in R(N, x, y)$, we build a lattice $\mathcal{L}(N, x, y, r)$. This lattice is derived from the target side of the rule α^r by concatenating lattices

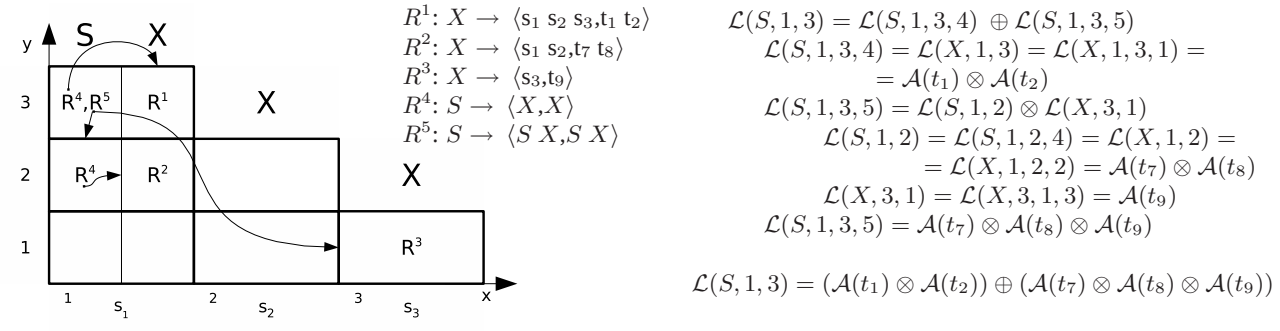


Figure 1: Production of target lattice $\mathcal{L}(S,1,3)$ using translation rules within CYK grid for sentence $s_1 s_2 s_3$. The grid is represented here in two dimensions (x, y). In practice only the first column accepts both non-terminals (S, X). For this reason it is divided in two subcolumns.

corresponding to the elements of $\alpha^r = \alpha^r_1 ... \alpha^r_{|\alpha^r|}$. If an α^r_i is a terminal, creating its lattice is straightforward. If α^r_i is a non-terminal, it refers to a cell (N', x', y') lower in the grid identified by the backpointer $BP(N, x, y, r, i)$; in this case, the lattice used is $\mathcal{L}(N', x', y')$. Taken together,

$$\mathcal{L}(N,x,y,r) = \bigotimes_{i=1..|\alpha^r|} \mathcal{L}(N,x,y,r,i) \quad (1)$$

$$\mathcal{L}(N,x,y,r,i) = \begin{cases} \mathcal{A}(\alpha_i) & \text{if } \alpha_i \in \mathbf{T} \\ \mathcal{L}(N',x',y') & \text{else} \end{cases} \quad (2)$$

where $\mathcal{A}(t)$, $t \in \mathbf{T}$ returns a single-arc acceptor which accepts only the symbol t. The lattice $\mathcal{L}(N, x, y)$ is then built as the union of lattices corresponding to the rules in $R(N, x, y)$:

$$\mathcal{L}(N,x,y) = \bigoplus_{r \in R(N,x,y)} \mathcal{L}(N,x,y,r) \quad (3)$$

Lattice union and concatenation are performed using the $\oplus$ and $\otimes$ WFST operations respectively, as described by Allauzen et al.(2007). If a rule R^r has a cost c^r, it is applied to the exit state of the lattice $L(N, x, y, r)$ prior to the operation of Equation 3.

2.1.1 An Example of Phrase-based Translation

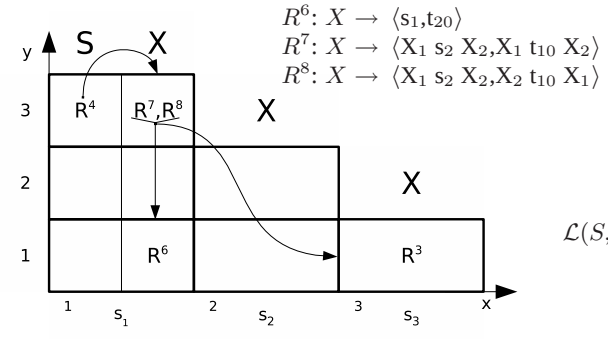
Figure 1 illustrates this process for a three word source sentence $s_1 s_2 s_3$ under monotone phrase-based translation. The left-hand side shows the state of the CYK grid after parsing using the rules R^1 to R^5. These include 3 rules with only terminals (R^1, R^2, R^3) and the glue rules (R^4, R^5). Arrows represent backpointers to lower-level cells. We are interested in the upper-most S cell $(S, 1, 3)$, as it represents the search space of translation hypotheses covering the whole source sentence. Two rules (R^4, R^5) are in this cell, so the lattice $\mathcal{L}(S, 1, 3)$ will be obtained by the union of the two lattices found by the backpointers of these two rules. This process is explicitly derived in the right-hand side of Figure 1.

2.1.2 An Example of Hierarchical Translation

Figure 2 shows a hierarchical scenario for the same sentence. Three rules, R^6, R^7, R^8, are added to the example of Figure 1, thus providing two additional derivations. This makes use of sublattices already produced in the creation of $\mathcal{L}(S, 1, 3, 5)$ and $\mathcal{L}(X, 1, 3, 1)$ in Figure 1; these are within $\{\}$.

2.2 A Procedure for Lattice Construction

Figure 3 presents an algorithm to build the lattice for every cell. The algorithm uses memoization: if a lattice for a requested cell already exists, it is returned (line 2); otherwise it is constructed via equations 1,2,3. For every rule, each element of the target side (lines 3,4) is checked as terminal or non-terminal (equation 2). If it is a terminal element (line 5), a simple acceptor is built. If it is a non-terminal (line 6), the lattice associated to its backpointer is returned (lines 7 and 8). The complete lattice $\mathcal{L}(N, x, y, r)$ for each rule is built by equation 1 (line 9). The lattice $\mathcal{L}(N, x, y)$ for this cell is then found by union of all the component rules (line 10, equation 3); this lattice is then reduced by



$$\mathcal{L}(S,1,3) = \mathcal{L}(S,1,3,4) \oplus \{\mathcal{L}(S,1,3,5)\}$$
$$\mathcal{L}(S,1,3,4) = \mathcal{L}(X,1,3) =$$
$$= \{\mathcal{L}(X,1,3,1)\} \oplus \mathcal{L}(X,1,3,7) \oplus \mathcal{L}(X,1,3,8)$$
$$\mathcal{L}(X,1,3,7) = \mathcal{L}(X,1,1,6) \otimes \mathcal{A}(t_{10}) \otimes \mathcal{L}(X,3,1,3) =$$
$$= \mathcal{A}(t_{20}) \otimes \mathcal{A}(t_{10}) \otimes \mathcal{A}(t_9)$$
$$\mathcal{L}(X,1,3,8) = \mathcal{A}(t_9) \otimes \mathcal{A}(t_{10}) \otimes \mathcal{A}(t_{20})$$

$$\mathcal{L}(S,1,3) = \{(\mathcal{A}(t_1) \otimes \mathcal{A}(t_2))\} \oplus$$
$$\oplus(\mathcal{A}(t_{20}) \otimes \mathcal{A}(t_{10}) \otimes \mathcal{A}(t_9)) \oplus (\mathcal{A}(t_9) \otimes \mathcal{A}(t_{10}) \otimes \mathcal{A}(t_{20})) \oplus$$
$$\oplus\{(\mathcal{A}(t_7) \otimes \mathcal{A}(t_8) \otimes \mathcal{A}(t_9))\}$$

Figure 2: Translation as in Figure 1 but with additional rules R^6,R^7,R^8. Lattices previously derived appear within $\{\}$.

standard WFST operations (lines 11,12,13). It is important at this point to remove any epsilon arcs which may have been introduced by the various WFST union, concatenation, and replacement operations (Allauzen et al., 2007).

1 **function** buildFst(N,x,y)
2 if $\exists\ \mathcal{L}(N,x,y)$ return $\mathcal{L}(N,x,y)$
3 for $r \in R(N,x,y)$, $R^r : N \rightarrow \langle \gamma, \alpha \rangle$
4 for $i = 1...|\alpha|$
5 if $\alpha_i \in \mathbf{T}$, $\mathcal{L}(N,x,y,r,i) = \mathcal{A}(\alpha_i)$
6 else
7 $(N',x',y') = BP(\alpha_i)$
8 $\mathcal{L}(N,x,y,r,i) =$ buildFst(N',x',y')
9 $\mathcal{L}(N,x,y,r) = \bigotimes_{i=1..|\alpha|} \mathcal{L}(N,x,y,r,i)$
10 $\mathcal{L}(N,x,y) = \bigoplus_{r \in R(N,x,y)} \mathcal{L}(N,x,y,r)$
11 fstRmEpsilon $\mathcal{L}(N,x,y)$
12 fstDeterminize $\mathcal{L}(N,x,y)$
13 fstMinimize $\mathcal{L}(N,x,y)$
14 return $\mathcal{L}(N,x,y)$

Figure 3: Recursive Lattice Construction.

2.3 Delayed Translation

Equation 2 leads to the recursive construction of lattices in upper-levels of the grid through the union and concatenation of lattices from lower levels. If equations 1 and 3 are actually carried out over fully expanded word lattices, the memory required by the upper lattices will increase exponentially.

To avoid this, we use special arcs that serve as pointers to the low-level lattices. This effectively builds a skeleton of the desired lattice and delays the creation of the final word lattice until a single replacement operation is carried out in the top cell $(S,1,J)$. To make this exact, we define a function $g(N,x,y)$ which returns a unique tag for each lattice in each cell, and use it to redefine equation 2. With the backpointer $(N',x',y') = BP(N,x,y,r,i)$, these special arcs are introduced as:

$$\mathcal{L}(N,x,y,r,i) = \begin{cases} \mathcal{A}(\alpha_i) & \text{if } \alpha_i \in \mathbf{T} \\ \mathcal{A}(g(N',x',y')) & \text{else} \end{cases} \quad (4)$$

The resulting lattices $\mathcal{L}(N,x,y)$ are a mix of target language words and lattice pointers (Figure 4, top). However each still represents the entire search space of all translation hypotheses covering the span. Importantly, operations on these lattices – such as lossless size reduction via determinization and minimization – can still be performed. Owing to the existence of multiple hierarchical rules which share the same low-level dependencies, these operations can greatly reduce the size of the skeleton lattice; Figure 4 shows the effect on the translation example. This process is carried out for the lattice at every cell, even at the lowest level where there are only sequences of word terminals. As stated, size reductions can be significant. However not all redudancy is removed, since duplicate paths may arise through the concatenation and union of sublattices with different spans.

At the upper-most cell, the lattice $\mathcal{L}(S,1,J)$ contains pointers to lower-level lattices. A single FST replace operation (Allauzen et al., 2007) recursively substitutes all pointers by their lower-level lattices until no pointers are left, thus producing the complete target word lattice for the whole source sentence. The use of the lattice pointer arc was inspired by the 'lazy evaluation' techniques developed by Mohri et al (2000). Its implementation uses the infrastructure provided by the OpenFST libraries for

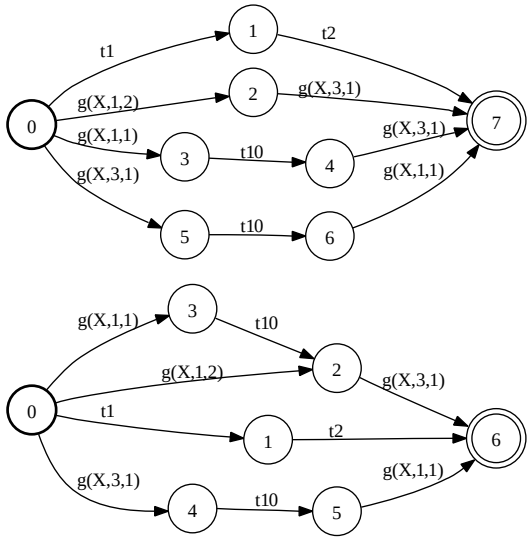


Figure 4: Delayed translation WFST with derivations from Figure 1 and Figure 2 before [t] and after minimization [b].

delayed composition, etc.

2.4 Pruning in Lattice Construction

The final translation lattice $\mathcal{L}(S, 1, J)$ can grow very large after the pointer arcs are expanded. We therefore apply a word-based language model, via WFST composition, and perform likelihood-based pruning (Allauzen et al., 2007) based on the combined translation and language model scores.

Pruning can also be performed on sublattices during search. One simple strategy is to monitor the number of states in the determinized lattices $\mathcal{L}(N, x, y)$. If this number is above a threshold, we expand any pointer arcs and apply a word-based language model via composition. The resulting lattice is then reduced by likelihood-based pruning, after which the LM scores are removed. This search pruning can be very selective. For example, the pruning threshold can depend on the height of the cell in the grid. In this way the risk of search errors can be controlled.

3 Translation Experiments

We report experiments on the NIST MT08 Arabic-to-English and Chinese-to-English translation tasks. We contrast two hierarchical phrase-based decoders. The first decoder, Hiero Cube Pruning (HCP), is a k-best decoder using cube pruning implemented as described by Chiang (2007). In our implementation, k-best lists contain unique hypotheses. The second decoder, Hiero FST (HiFST), is a lattice-based decoder implemented with Weighted Finite State Transducers as described in the previous section. Hypotheses are generated after determinization under the tropical semiring so that scores assigned to hypotheses arise from single minimum cost / maximum likelihood derivations. We also use a variant of the k-best decoder which works in alignment mode: given an input k-best list, it outputs the feature scores of each hypothesis in the list without applying any pruning. This is used for Minimum Error Training (MET) with the HiFST system.

These two language pairs pose very different translation challenges. For example, Chinese-to-English translation requires much greater word movement than Arabic-to-English. In the framework of hierarchical translation systems, we have found that shallow decoding (see section 3.2) is as good as full hierarchical decoding in Arabic-to-English (Iglesias et al., 2009). In Chinese-to-English, we have not found this to be the case. Therefore, we contrast the performance of HiFST and HCP under shallow hierarchical decoding for Arabic-to-English, while for Chinese-to-English we perform full hierarchical decoding.

Both hierarchical translation systems share a common architecture. For both language pairs, alignments are generated over the parallel data. The following features are extracted and used in translation: target language model, source-to-target and target-to-source phrase translation models, word and rule penalties, number of usages of the glue rule, source-to-target and target-to-source lexical models, and three rule count features inspired by Bender et al. (2007). The initial English language model is a 4-gram estimated over the parallel text and a 965 million word subset of monolingual data from the English Gigaword Third Edition. Details of the parallel corpus and development sets used for each language pair are given in their respective section.

Standard MET (Och, 2003) iterative parameter estimation under IBM BLEU (Papineni et al., 2001) is performed on the corresponding development set. For the HCP system, MET is done following Chiang (2007). For the HiFST system, we obtain a k-

best list from the translation lattice and extract each feature score with the aligner variant of the k-best decoder. After translation with optimized feature weights, we carry out the two following rescoring steps.

- *Large-LM rescoring*. We build sentence-specific zero-cutoff stupid-backoff (Brants et al., 2007) 5-gram language models, estimated using $\sim$4.7B words of English newswire text, and apply them to rescore either 10000-best lists generated by HCP or word lattices generated by HiFST. Lattices provide a vast search space relative to k-best lists, with translation lattice sizes of 10^{81} hypotheses reported in the literature (Tromble et al., 2008).

- *Minimum Bayes Risk (MBR)*. We rescore the first 1000-best hypotheses with MBR, taking the negative sentence level BLEU score as the loss function (Kumar and Byrne, 2004).

3.1 Building the Rule Sets

We extract hierarchical phrases from word alignments, applying the same restrictions as introduced by Chiang (2005). Additionally, following Iglesias et al. (2009) we carry out two rule filtering strategies:

- we exclude rules with two non-terminals with the same order on the source and target side

- we consider only the 20 most frequent translations for each rule

For each development set, this produces approximately 4.3M rules in Arabic-to-English and 2.0M rules in Chinese-to-English.

3.2 Arabic-to-English Translation

We translate Arabic-to-English with shallow hierarchical decoding, *i.e.* only phrases are allowed to be substituted into non-terminals. The rules used in this case are, in addition to the glue rules:

$$X \rightarrow \langle \gamma_s, \alpha_s \rangle$$
$$X \rightarrow \langle V, V \rangle$$
$$V \rightarrow \langle s, t \rangle$$
$$s, t \in \mathbf{T}^{+}; \gamma_s, \alpha_s \in (\{V\} \cup \mathbf{T})^{+}$$

For translation model training, we use all allowed parallel corpora in the NIST MT08 Arabic track ($\sim$150M words per language). In addition to the MT08 set itself, we use a development set *mt02-05-tune* formed from the odd numbered sentences of the NIST MT02 through MT05 evaluation sets; the even numbered sentences form the validation set *mt02-05-test*. The *mt02-05-tune* set has 2,075 sentences.

The cube pruning decoder, HCP, employs k-best lists of depth k=10000 (unique). Using deeper lists results in excessive memory and time requirements. In contrast, the WFST-based decoder, HiFST, requires no local pruning during lattice construction for this task and the language model is not applied until the lattice is fully built at the upper-most cell of the CYK grid.

Table 1 shows results for *mt02-05-tune*, *mt02-05-test* and *mt08*, as measured by lowercased IBM BLEU and TER (Snover et al., 2006). MET parameters are optimized for the HCP decoder. As shown in rows 'a' and 'b', results after MET are comparable.

Search Errors Since both decoders use exactly the same features, we can measure their search errors on a sentence-by-sentence basis. A search error is assigned to one of the decoders if the other has found a hypothesis with lower cost. For *mt02-05-tune*, we find that in 18.5% of the sentences HiFST finds a hypothesis with lower cost than HCP. In contrast, HCP never finds any hypothesis with lower cost for any sentence. This is as expected: the HiFST decoder requires no pruning prior to applying the language model, so search is exact.

Lattice/k-best Quality Rescoring results are different for cube pruning and WFST-based decoders. Whereas HCP improves by 0.9 BLEU, HiFST improves over 1.5 BLEU. Clearly, search errors in HCP not only affect the 1-best output but also the quality of the resulting k-best lists. For HCP, this limits the possible gain from subsequent rescoring steps such as large LMs and MBR.

Translation Speed HCP requires an average of 1.1 seconds per input word. HiFST cuts this time by half, producing output at a rate of 0.5 seconds per word. It proves much more efficient to process compact lattices contaning many hypotheses rather than to independently processing each one of them in k-best form.

	decoder	*mt02-05-tune*		*mt02-05-test*		*mt08*	
		BLEU	TER	BLEU	TER	BLEU	TER
a	HCP	52.2	41.6	51.5	42.2	42.5	48.6
	+5gram	53.1	41.0	52.5	41.5	43.3	48.3
	+MBR	53.2	40.8	52.6	41.4	43.4	48.1
b	HiFST	52.2	41.5	51.6	42.1	42.4	48.7
	+5gram	53.3	40.6	52.7	41.3	43.7	48.1
	+MBR	53.7	40.4	53.3	40.9	44.0	48.0
Decoding time in secs/word: 1.1 for HCP; 0.5 for HiFST.							

Table 1: Constrative Arabic-to-English translation results (lower-cased IBM BLEU | TER) after MET and subsequent rescoring steps. Decoding time reported for *mt02-05-tune*.

The mixed case NIST BLEU-4 for the HiFST system on *mt08* is 42.9. This is directly comparable to the official MT08 Constrained Training Track evaluation results[1].

3.3 Chinese-to-English Translation

We translate Chinese-to-English with full hierarchical decoding, *i.e.* hierarchical rules are allowed to be substituted into non-terminals. We consider a maximum span of 10 words for the application of hierarchical rules and only glue rules are allowed at upper levels of the CYK grid.

For translation model training, we use all available data for the GALE 2008 evaluation[2], approx. 250M words per language. In addition to the MT08 set itself, we use a development set *tune-nw* and a validation set *test-nw*. These contain a mix of the newswire portions of MT02 through MT05 and additional developments sets created by translation within the GALE program. The *tune-nw* set has 1,755 sentences.

Again, the HCP decoder employs k-best lists of depth k=10000. The HiFST decoder applies pruning in search as described in Section 2.4, so that any lattice in the CYK grid is pruned if it covers at least 3 source words and contains more than 10k states. The likelihood pruning threshold relative to the best path in the lattice is 9. This is a very broad threshold so that very few paths are discarded.

[1]Full MT08 results are available at http://www.nist.gov/speech/tests/mt/2008/doc/mt08_official_results_v0.html. It is worth noting that many of the top entries make use of system combination; the results reported here are for single system translation.

[2]See http://projects.ldc.upenn.edu/gale/data/catalog.html.

Improved Optimization Table 2 shows results for *tune-nw*, *test-nw* and *mt08*, as measured by lower-cased IBM BLEU and TER. The first two rows show results for HCP when using MET parameters optimized over k-best lists produced by HCP (row 'a') and by HiFST (row 'b'). We find that using the k-best list obtained by the HiFST decoder yields better parameters during optimization. Tuning on the HiFST k-best lists improves the HCP BLEU score, as well. We find consistent improvements in BLEU; TER also improves overall, although less consistently.

Search Errors Measured over the *tune-nw* development set, HiFST finds a hypothesis with lower cost in 48.4% of the sentences. In contrast, HCP never finds any hypothesis with a lower cost for any sentence, indicating that the described pruning strategy for HiFST is much broader than that of HCP. Note that HCP search errors are more frequent for this language pair. This is due to the larger search space required in fully hierarchical translation; the larger the search space, the more search errors will be produced by the cube pruning k-best implementation.

Lattice/k-best Quality The lattices produced by HiFST yield greater gains in LM rescoring than the k-best lists produced by HCP. Including the subsequent MBR rescoring, translation improves as much as 1.2 BLEU, compared to 0.7 BLEU with HCP. The mixed case NIST BLEU-4 for the HiFST system on *mt08* is 27.8, comparable to official results in the UnConstrained Training Track of the NIST 2008 evaluation.

	decoder	MET k-best	*tune-nw*		*test-nw*		*mt08*	
			BLEU	TER	BLEU	TER	BLEU	TER
a	HCP	HCP	31.6	59.7	31.9	59.7	–	–
b	HCP	HiFST	31.7	60.0	32.2	59.9	27.2	60.2
	+5gram		32.2	59.3	32.6	59.4	27.8	59.3
	+MBR		32.4	59.2	32.7	59.4	28.1	59.3
c	HiFST	HiFST	32.0	60.1	32.2	60.0	27.1	60.5
	+5gram		32.7	58.3	33.1	58.4	28.1	59.1
	+MBR		32.9	58.4	33.4	58.5	28.9	58.9

Table 2: Contrastive Chinese-to-English translation results (lower-cased IBM BLEU|TER) after MET and subsequent rescoring steps. The MET k-best column indicates which decoder generated the k-best lists used in MET optimization.

4 Conclusions

The lattice-based decoder for hierarchical phrase-based translation described in this paper can be easily implemented using Weighted Finite State Transducers. We find many benefits in this approach to translation. From a practical perspective, the computational operations required can be easily carried out using standard operations already implemented in general purpose libraries. From a modeling perspective, the compact representation of multiple translation hypotheses in lattice form requires less pruning in hierarchical search. The result is fewer search errors and reduced overall memory use relative to cube pruning over k-best lists. We also find improved performance of subsequent rescoring procedures which rely on the translation scores. In direct comparison to k-best lists generated under cube pruning, we find that MET parameter optimization, rescoring with large language models, and MBR decoding, are all improved when applied to translations generated by the lattice-based hierarchical decoder.

Acknowledgments

This work was supported in part by the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011- 06-C-0022. G. Iglesias supported by Spanish Government research grant BES-2007-15956 (project TEC2006-13694-C03-03).

References

Cyril Allauzen, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of ACL*, pages 557–564.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of CIAA*, pages 11–23.

Srinivas Bangalore and Giuseppe Riccardi. 2001. A finite-state approach to machine translation. In *Proceedings of NAACL*.

Oliver Bender, Evgeny Matusov, Stefan Hahn, Sasa Hasan, Shahram Khadivi, and Hermann Ney. 2007. The RWTH Arabic-to-English spoken language translation system. In *Proceedings of ASRU*, pages 396–401.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-HLT*, pages 200–208.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP-ACL*, pages 858–867.

Francisco Casacuberta. 2001. Finite-state transducers for speech-input translation. In *Proceedings of ASRU*.

Jean-Cédric Chappelier and Martin Rajman. 1998. A generalized CYK algorithm for parsing stochastic CFG. In *Proceedings of TAPD*, pages 133–137.

Jean-Cédric Chappelier, Martin Rajman, Ramón Aragüés, and Antoine Rozenknop. 1999. Lattice parsing for speech recognition. In *Proceedings of TALN*, pages 95–104.

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

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL-HLT*, pages 1012–1020.

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

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of ACL*, pages 144–151.

Gonzalo Iglesias, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009. Rule filtering by pattern for efficient hierarchical translation. In *Proceedings of EACL*.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of HLT-NAACL*, pages 169–176.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of HLT-EMNLP*, pages 161–168.

Zhifei Li and Sanjeev Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of the ACL-HLT Second Workshop on Syntax and Structure in Statistical Translation*, pages 10–18.

Adam Lopez. 2008. Tera-scale translation models via pattern matching. In *Proceedings of COLING*, pages 505–512.

Yuval Marton and Philip Resnik. 2008. Soft syntactic constraints for hierarchical phrased-based translation. In *Proceedings of ACL-HLT*, pages 1003–1011.

Lambert Mathias and William Byrne. 2006. Statistical phrase-based speech translation. In *Proceedings of ICASSP*.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231:17–32.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. In *Computer Speech and Language*, volume 16, pages 69–88.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr. 2007. Combining outputs from multiple machine translation systems. In *Proceedings of HLT-NAACL*, pages 228–235.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-HLT*, pages 577–585.

Khe Chai Sim, William Byrne, Mark Gales, Hichem Sahbi, and Phil Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *Proceedings of ICASSP*, volume 4, pages 105–108.

Matthew Snover, Bonnie J. Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*, pages 223–231.

Roy Tromble, Shankar Kumar, Franz J. Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *Proceedings of EMNLP*, pages 620–629.

Ashish Venugopal, Andreas Zollmann, and Vogel Stephan. 2007. An efficient two-pass approach to synchronous-CFG driven statistical MT. In *Proceedings of HLT-NAACL*, pages 500–507.

Hao Zhang and Daniel Gildea. 2006. Synchronous binarization for machine translation. In *Proceedings of HLT-NAACL*, pages 256–263.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of NAACL Workshop on Statistical Machine Translation*, pages 138–141.

Andreas Zollmann, Ashish Venugopal, Franz Och, and Jay Ponte. 2008. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of COLING*, pages 1145–1152.

Improved Pronunciation Features for Construct-driven Assessment of Non-native Spontaneous Speech

Lei Chen, Klaus Zechner, Xiaoming Xi
Educational Testing Service
Princeton, NJ, USA
{LChen,KZechner,XXi}@ets.org

Abstract

This paper describes research on automatic assessment of the pronunciation quality of spontaneous non-native adult speech. Since the speaking content is not known prior to the assessment, a two-stage method is developed to first recognize the speaking content based on non-native speech acoustic properties and then forced-align the recognition results with a reference acoustic model reflecting native and near-native speech properties. Features related to Hidden Markov Model likelihoods and vowel durations are extracted. Words with low recognition confidence can be excluded in the extraction of likelihood-related features to minimize erroneous alignments due to speech recognition errors. Our experiments on the TOEFL®Practice Online test, an English language assessment, suggest that the recognition/forced-alignment method can provide useful pronunciation features. Our new pronunciation features are more meaningful than an utterance-based normalized acoustic model score used in previous research from a construct point of view.

1 Introduction

Automated systems for evaluating highly predictable speech (e.g. read speech or speech that is quite constrained in the use of vocabulary and syntactic structures) have emerged in the past decade (Bernstein, 1999; Witt, 1999; Franco et al., 2000; Hacker et al., 2005) due to the growing maturity of speech recognition and processing technologies. However, endeavors into automated scoring for spontaneous speech have been sparse given the challenge of both recognizing and assessing spontaneous speech. This paper addresses the development and evaluation of pronunciation features for an automated system for scoring spontaneous speech. This system was deployed for the TOEFL®Practice Online (TPO) assessment used by prospective test takers to prepare for the official TOEFL®test.

A construct is a set of knowledge, skills, and abilities measured by a test. The construct of the speaking test is embodied in the rubrics that human raters use to score the test. It consists of three key categories: delivery, language use, and topic development. Delivery refers to the pace and the clarity of the speech, including performance on intonation, rhythm, rate of speech, and degree of hesitancy. Language use refers to the range, complexity, and precision of vocabulary and grammar use. Topic development refers to the coherence and fullness of the response. Most of the research on spontaneous speech assessment focuses on the delivery aspect given the low recognition accuracy on non-native spontaneous speech.

The delivery aspect can be measured on four dimensions: fluency, intonation, rhythm, and pronunciation. For the TPO assessment, we have defined pronunciation as the quality of vowels, consonants and word-level stress (segmentals). Intonation and sentence-level stress patterns (supra-segmentals) are not defined as part of pronunciation. Pronunciation is one of the key factors that impact the intelligibility and perceived comprehensibility of speech. Because pronunciation plays an important role in speech perception, features measuring pronuncia-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 442–449,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

tion using speech technologies have been explored in many previous studies. However, the bulk of the research on automatic pronunciation evaluation concerns read speech or highly predictable speech (Witt, 1999; Franco et al., 2000; Hacker et al., 2005), where there is a high possibility of success in speech recognition. Automatic pronunciation evaluation is challenging for spontaneous speech and has been under-explored.

In this paper, we will describe a method for extracting pronunciation features based on spontaneous speech that is well motivated by theories and supported by empirical evaluations of feature performance. In conceptualizing and computing these features, we draw on the literature on automatic pronunciation evaluation for constrained speech. As described in the related work in Section 2, the widely used features for measuring pronunciation are (1) likelihood (posterior probability) of a phoneme being spoken given the observed audio sample that is computed in a Viterbi decoding process, and (2) phoneme length measurements that are compared to standard references based on native speech.

However, we have also come up with unique solutions to address the issue of relatively low accuracy in recognizing spontaneous speech. Our methods of feature extraction are designed with considerations of how to best capture the quality of pronunciation given technological constraints.

The remainder of the paper is organized as follows: Section 2 reviews the related research; Section 3 describes our method to extract a set of features for measuring pronunciation; Section 4 describes the design of the experiments, including the questions investigated, the data, the speech processing technologies, and the measurement metrics; Section 5 reports on the experimental results; Section 6 discusses the experimental results; and Section 7 summaries the findings and future research planned.

2 Related work

There is previous research on utilizing speech recognition technology to automatically assess non-native speakers' communicative competence (e.g., fluency, intonation, and pronunciation). Witt (Witt, 1999) developed the Goodness of Pronunciation (GOP) measurement for measuring pronunciation based on Hidden Markov Model (HMM) log likelihood. Using a similar method, Neumeyer et al. (Neumeyer et al., 2000) designed a series of likelihood related pronunciation features, e.g., the local average likelihood and global average likelihood. Hacker et al. (Hacker et al., 2005) utilized a relatively large feature vector for scoring pronunciation.

Pronunciation has been the focus of assessment in several automatic speech scoring systems. Franco et al. (Franco et al., 2000) presented a system for automatic evaluation of pronunciation quality on the phoneme level and the sentence level of speech by native and non-native speakers of English and other languages (e.g., French). A forced alignment between the speech read by subjects and the ideal path through the HMM was computed. Then, the log posterior probabilities for a certain position in the signal were computed to achieve a local pronunciation score. Cucchiarini et al. (Cucchiarini et al., 1997a; Cucchiarini et al., 1997b) designed a system for scoring Dutch pronunciation along a similar line. Their pronunciation feature set was more extensive, including various log likelihood HMM scores and phoneme duration scores. In these two systems, the speaking skill scores computed on features by machine are found to have good agreement with scores provided by humans.

A limited number of studies have been conducted on assessing speaking proficiency based on spontaneous speech. Moustroufas and Digalakis (Moustroufas and Digalakis, 2007) designed a system to automatically evaluate the pronunciation of foreign speakers using unknown text. The difference in the recognition results between a recognizer trained on speakers' native languages (L1) and another recognizer trained on their learned languages (L2) was used for pronunciation scoring. Zechner and Bejar (Zechner and Bejar, 2006) presented a system to score non-native spontaneous speech using features derived from the recognition results. Following their work, an operational assessment system, SpeechRater^{TM}, was implemented with further improvements (Zechner et al., 2007).

There are some issues with the method to extract pronunciation features in the previous research on automated assessment of spontaneous speech (Zechner and Bejar, 2006; Zechner et al., 2007). For ex-

ample, the acoustic model (AM) that was used to estimate a likelihood of a phoneme being spoken was well-fitted to non-native speech acoustic properties. Further, other important aspects of pronunciation, e.g., vowel duration, have not been utilized as a feature in the current SpeechRater$^{\text{TM}}$ system. Likelihoods estimated on non-words (such as silences and fillers) that were not central to the measurement of pronunciation were used in the feature extraction. In addition, mis-recognized words lead to wrong likelihood estimation. Our paper attempts to address all of these limitations described above.

3 Extraction of Pronunciation Features

Figure 1 depicts our new method for extracting an expanded set of pronunciation features in a more meaning way.

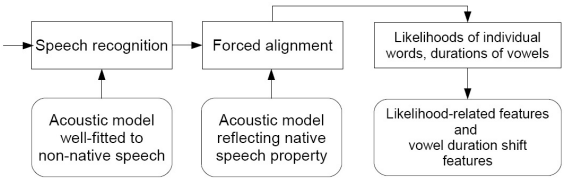


Figure 1: Two-stage pronunciation feature extraction

We used two different AMs for pronunciation feature extraction. First, we used an AM optimized for speech recognition (typically an AM adapted on non-native speech to better fit non-native speakers' acoustics patterns) to generate word hypotheses; then we used the other AM optimized for pronunciation scoring (typically trained on native or near-native speech to be a good reference model reflecting expected speech characteristics) to force align the speech signals to the word hypotheses and to compute the likelihoods of individual words being spoken and durations of phonemes; finally new pronunciation features were extracted based on these measurements.

Some notations used for computing the pronunciation features are listed in Table 1. Based on these notations, the proposed new pronunciation features are described in Table 2. To address the limitations of previous research on automated assessment of pronunciation, which was described in Section 2, our proposed method has achieved improvements on (1) using the two-stage method to compute HMM likelihoods using a reference acoustic model trained on native and near-native speech, (2) expanding the coverage of pronunciation features by using vowel duration shifts that are compared to standard norms of native speech, (3) and using likelihoods on the audio portions that are recognized as words and applying various normalizations.

Table 1: Notations used for pronunciation feature extraction

Variable	Meaning
$L(x_i)$	the likelihood of word x_i being spoken given the observed audio signal
t_i	the duration of word i in a response
T_s	the duration of the entire response
T	$\sum_{i=1}^{n} t_i$, the summation of the duration of all words, where $T \leq T_s$
n	the number of words in a response
m	the number of letters in a response
R	$\frac{m}{T_s}$, the frequency of letters (as the rate of speech)
v_i	vowel i
N_v	the total number of vowels
P_{v_i}	the duration of vowel v_i
$\bar{P}$	the average vowel duration (across all vowels in the response being scored)
D_{v_i}	the standard average duration of vowel v_i (estimated on a native speech corpus)
$\bar{D}$	the averaged vowel duration (on all vowels in a native speech corpus)
S_{v_i}	$\lvert P_{v_i} - D_{v_i} \rvert$, duration shift of vowel v_i (measured as the absolute value of the difference between the duration of vowel v_i and its standard value)
Sn_{v_i}	$\lvert \frac{P_{v_i}}{\bar{P}} - \frac{D_{v_i}}{\bar{D}} \rvert$, normalized duration shift of vowel v_i (measured as the absolute value of the normalized difference between the duration of vowel v_i and its standard value)

4 Experiment design

We first raise three questions that we try to answer with our experiments. Then, we describe the data sets and the speech recognizers, especially the two

Table 2: A list of proposed pronunciation features

Feature	Formula	Meaning
L_1	$\sum_{i=1}^{n} L(x_i)$	summation of likelihoods of all the individual words
L_2	L_1/n	average likelihood across all words
L_3	L_1/m	average likelihood across all letters
L_4	L_1/T	average likelihood per second
L_5	$\frac{\sum_{i=1}^{n} \frac{L(x_i)}{t_i}}{n}$	average likelihood density across all words
L_6	L_4/R	L_4 normalized by the rate of speech
L_7	L_5/R	L_5 normalized by the rate of speech
$\bar{S}$	$\frac{\sum_{i=1}^{N_v} S_{v_i}}{N_v}$	average vowel duration shifts
$\bar{Sn}$	$\frac{\sum_{i=1}^{N_v} Sn_{v_i}}{N_v}$	average normalized vowel duration shifts

different acoustic models fitted to non-native and expected speech respectively. Finally, we describe the evaluation criterion used in the experiment.

4.1 Research questions

In order to justify that the two-stage method for extracting pronunciation features is a valid method that provides useful features for assessing pronunciation, the following questions need to be answered:

Q1: Can the words hypothesized be used to approximate the human transcripts in the forced alignment step?

Q2: Are the new pronunciation features effective for assessment?

Q3: Can the likelihood-related features be improved when using only words correctly recognized?

4.2 Data

Table 3 lists the data sets used in the experiment. Non-native speech collected in the TPO was used in training a non-native AM. For feature evaluations, we selected $1,257$ responses from the TPO data collected in 2006. Within this set, 645 responses were transcribed. Holistic scores were assigned by human raters based on a score scale of 1 (the lowest proficiency) to 4 (the highest proficiency).

In the TOEFL®Native Speaker Study, native speakers of primarily North American English (NaE) took the TOEFL®test and their speech files were collected. This TOEFL®native speech data and some high-scored TPO responses were used in the adaptation of an AM representing expected speech properties. In addition, $1,602$ responses of native speech, which had the highest speech proficiency scores in NaE, were used to estimate standard average vowel durations.

Type	Function	Source	Size
non-native speech	AM training feature evaluation	TPO TPO collected in 2006	$\sim$ 30 hrs $1,257$ responses (645 with transcripts)
native or near-native speech	AM adaptation estimation of standard vowel durations	TPO and TOEFL Native TOEFL Native	$\sim 2,000$ responses $1,602$ responses

Table 3: Data sets used in the experiment

4.3 Speech technologies

For speech recognition and forced alignment, we used a gender-independent fully continuous HMM speech recognizer. Two different AMs were used in the recognition and forced alignment steps respectively.

The AM used in the recognition was trained on about 30 hours of non-native speech from the TPO. For language model training, a large corpus of non-native speech (about 100 hours) was used

and mixed with a large general-domain language model (trained from the Broadcast News (BN) corpus (Graff et al., 1997) of the Linguistic Data Consortium (LDC)). In the pronunciation feature extraction process depicted in Figure 1, this AM was used to recognize non-native speech to generate the word hypotheses.

The AM used in the forced alignment was trained on native speech and high-scored non-native speech. It was trained as follows: starting from a generic recognizer, which was trained on a large and varied native speech corpus, we adapted the AM using batch-mode MAP adaptation. The adaptation corpus contained about $2,000$ responses with high scores in previous TPO tests and the TOEFL®Native Speaker Study. In addition, this AM was used to estimate standard norms of vowels as described in Table 1.

4.4 Measurement metric

To measure the quality of the developed features, a widely used metric is the Pearson correlation (r) computed between the features and human scores. In previous studies, human holistic scores of perceived proficiency have been widely used in estimating the correlations. In our experiment, we will use the absolute value of Pearson correlation with human holistic scores ($|r|$) to evaluate the features. Given the close relationship between pronunciation quality and overall speech proficiency, $|r|$ is expected to approximate the strength of its relationship with the human pronunciation scores.

5 Experimental Results

5.1 Results for Q1

When assessing read speech, the transcription of the spoken content is known prior to the assessment and used to forced-align the speech for feature extraction. However, when assessing spontaneous speech, we do not know the spoken content and cannot provide a correct word transcription for the forced alignment with imperfect speech recognition. A practical solution is to use the recognition hypothesis to approximate the human transcript in the forced alignment. Since the recognition word accuracy on non-native spontaneous speech is not very high (for example, a word accuracy of about 50% on the TPO data was reported in (Zechner et al., 2007)), it is critical to verify that the approximation can provide good enough pronunciation features compared to the ones computed in an ideal scenario (by using the human transcript in the forced alignment step).

We ran forced alignment on 645 TPO responses with human transcriptions, using both the manual transcription and the word hypotheses from the recognizer described in Section 4.3. Then, based on these two forced alignment outputs, we extracted the pronunciation features as described in Section 3.

Table 4 reports the $|r|$s between the proposed pronunciation features and human holistic scores when using the forced alignment results from either transcriptions or recognition hypotheses. The relative $|r|$ reduction (defined as $(|r|_{transcriptions} - |r|_{hypotheses})/|r|_{transcriptions} * 100$) is reported to measure the magnitude reduction.

Based on the results shown in Table 4, we find that the pronunciation features computed based on the forced alignment results using transcriptions have higher $|r|$s with the human holistic scores than the corresponding features computed based on the FA results using the recognition hypotheses. This is not surprising given that only $50\% \sim 60\%$ word accuracy can be achieved when recognizing non-native spontaneous speech. However, the pronunciation features computed using the recognition hypotheses that is feasible in practice show some promising correlations to human holistic scores. For example, L_3, L_6, and L_7 have $|r|$s larger than 0.45 and $\bar{Sn}$ has an $|r|$ larger than 0.35. Compared to the corresponding features computed using the FA results based on transcriptions, these promising pronunciation features that can be obtained practically, show some reduction in quality (from 13.4% to 21.1%) but are still usable. Therefore, our proposed two-stage method for pronunciation feature extraction is proven to be a practical way for the computation of features that have acceptable performance.

5.2 Result for Q2

Although our proposed modifications described in Section 3 have improved the meaningfulness of the features, an empirical study is needed to examine the actual utility of these features for the assessment of pronunciation.

In the experiment described in Section 5.1, four pronunciation features (including L_3, L_6, L_7, and

Feature	$\|r\|$ using transcription	$\|r\|$ using recognition hypothesis	relative $\|r\|$ reduction (%)
L_1	0.216	0.107	50.5
L_2	0.443	0.416	6.1
L_3	0.506	0.473	6.5
L_4	0.363	0.294	19
L_5	0.333	0.287	13.8
L_6	0.549	0.475	13.5
L_7	0.546	0.473	13.4
$\bar{S}$	0.396	0.296	25.3
$\bar{Sn}$	0.451	0.356	21.1

Table 4: $|r|$ between the pronunciation features and human holistic scores under two forced alignment input conditions (using transcriptions vs. using recognition hypotheses) and relative $|r|$ reduction

$\bar{Sn}$) show promising correlations to human holistic scores. To check the quality of the newly developed pronunciation features, we compared these four features with the *amscore* feature used in (Zechner et al., 2007) on the TPO data set collected in 2006 (with $1,257$ responses). We first ran speech recognition using the recognizer designed for non-native speech. The recognition results were used to compute the *amscore*, which is calculated by dividing the likelihood over an entire response by the number of letters. Then, we used the recognition hypotheses to do the forced alignment using the other AM trained on the native and near-native speech to extract those four pronunciation features. Finally, we calculated the correlation coefficients between features and the human holistic scores. The results are reported in Table 5.

feature	$\|r\|$ to human holistic scores
amscore	0.434
L_3	0.369
L_6	0.444
L_7	0.443
$\bar{Sn}$	0.363

Table 5: A comparison of new pronunciation features to *amscore*, the one used in SpeechRater$^{\text{TM}}$

Compared to the feature *amscore*, L_6 and L_7 have slightly higher $|r|$s with the human holistic scores. This suggests that our construct-driven approach yields pronunciation features that are empirically comparable or even better than the *amscore*. In addition, $\bar{Sn}$, a new feature representing the vowel production aspect of pronunciation, shows a relatively high correlation with human holistic scores. This suggests that our new pronunciation feature set has an expanded coverage of pronunciation.

It is interesting to note that L_3 has a lower $|r|$ with human holistic scores than the *amscore* does. Although the computation of L_3 is quite similar to that of *amscore*, the major difference is that likelihoods of non-word portions (such as silences and fillers) are used to compute *amscore* but not L_3. This suggests that likelihood-related pronunciation features that involve information related to non-words may perform better in predicting human holistic scores. For example, for *amscore*, the likelihoods measured on those non-word units were involved in the feature calculation; for L_6 and L_7, the temporal information of those non-word units (e.g., duration of units) was involved in the feature calculation [1].

5.3 Result for Q3

In the feature extraction, we used the words hypothesized by the speech recognizer as the input for the forced alignment. Since a considerable number of words are recognized incorrectly (especially for non-native spontaneous speech), a natural way to further improve the likelihood related features is to only consider words which are correctly recognized. A useful metric associated with the recognition performance is the confidence score (CS) output by the recognizer, which reflects the recognizer's estimation about the probability that a hypothesized word is correctly recognized. The recognized words with high confidence scores tend to be correctly recognized. Therefore, focusing on words recognized with high confidence scores may reduce the negative impact caused by recognition errors on the quality of the likelihood related features.

On the TPO data with human transcripts, we used the NIST's *sclite* scoring tool (Fiscus, 2009) to measure the percentage of correct words (correct%), which is defined as the ratio of the number of words

[1] L_6 and L_7 use R, which is computed as $\frac{m}{T_s}$, where T_s contains durations of non-words.

correctly recognized given the number of words in the reference transcript. On all words (corresponding to confidence scores ranging from 0.0 to 1.0), the correct% is 53.3%. Figure 2 depicts the correct% corresponding to ten confidence score bins ranging from 0.0 to 1.0. Clearly, with the increase of the confidence score, more words tend to be accurately recognized. Therefore, it is reasonable to only use likelihoods estimated on the hypothesized words with high confidence scores for extracting likelihood related features.

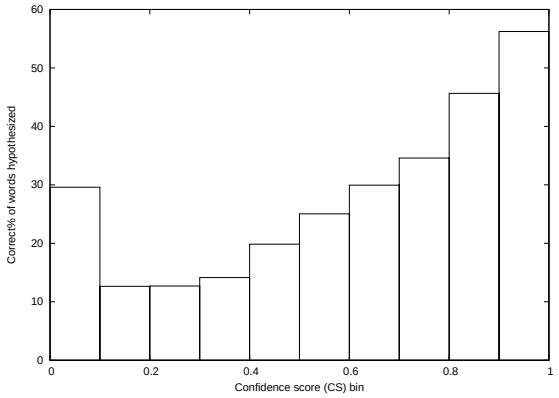


Figure 2: Correct% of words recognized across 10 confidence score bins

On the TPO data set collected in 2006, we computed three likelihood related features (including L_3, L_6, and L_7) only on words whose SC is equal to or higher than a threshold (i.e., 0.5, 0.6, 0.7, 0.8, and 0.9) and measured the $|r|$ of a feature with the human holistic scores. Table 6 lists the confidence score cutting thresholds, the percentage of words whose confidence scores are not lower than the cutting threshold selected, and $|r|$ between each likelihood feature to human holistic scores. In the Table 6, we observe that only using words recognized with high confidence improves the correlations between the features and the human holistic scores. One issue about only using words recognized with high confidence scores is that the number of words used in the feature extraction has been reduced and may reduce the robustness of the feature calculation.

T_c	percentage of words whose CS $\geq T_c$ (%)	L_3 $\|r\|$	L_6 $\|r\|$	L_7 $\|r\|$
0.0	100	0.369	0.444	0.443
0.5	84.21	0.38	0.462	0.461
0.6	77.07	0.377	0.465	0.464
0.7	69.31	0.363	0.461	0.461
0.8	60.86	0.371	0.466	0.466
0.9	50.76	0.426	0.477	0.475

Table 6: $|r|$ between L_3, L_6, and L_7 and human holistic scores using only words recognized whose CSs are not lower than a threshold (T_c)

6 Discussion

To assess the pronunciation of spontaneous speech, we proposed a method for extracting a set of pronunciation features. The method consists of two stages: (1) recognizing speech using an AM well fitted to non-native speech properties and (2) forced-aligning the hypothesized words using the other AM, which was trained on native and near-native speech, and extracting features related to spectral properties (HMM likelihood) and vowel production. This method of using one AM optimized for speech recognition and another AM optimized for pronunciation evaluation is well motivated theoretically. The derived pronunciation features have also been found to have reasonably high correlations with human holistic scores. The results support the linkage of the features to the construct of pronunciation and their utility of being used in a scoring model to predict human holistic judgments. Several contributions of this paper are described as below.

First, the two-stage method allows us to utilize an AM trained on native and near-native speech as a reference model when computing pronunciation features. The decision to include high-scored non-native speech was driven by the scoring rubrics derived from the construct, where the pronunciation quality of the highest level performance does not necessarily require native-like accent, but highly intelligible speech. The way the reference model was trained is consistent with the scoring rubrics, and makes it an appropriate standard based on which the pronunciation quality of non-native speech can be

evaluated. By using the recognition hypotheses from the recognition step as input in the forced alignment step, our experiments show a relatively small reduction in correlations with human holistic scores in comparison to the features based on the human transcriptions. This suggests that our method has potential to be implemented in a real-time operational setting.

Second, a few decisions we have made in computing the pronunciation features are driven by considerations of how these features are meaningfully linked to the construct of pronunciation assessment. For example, we have excluded the HMM likelihoods on non-words (such as pauses and fillers) in the computations of likelihood-related features. In addition, only using words recognized with high confidence scores yields more informative likelihood-related features for assessing the quality of speech. The inclusion of vowel duration measures in the feature set expanded the coverage of the quality of pronunciation.

7 Summary and future work

This paper presents a method for computing features for assessing the pronunciation quality of non-native spontaneous speech, guided by construct considerations. We were able to show that using a two-stage method of first recognizing speech with a non-native AM and then forced aligning of the hypothesis using a native or near-native speech AM we can generate pronunciation features with promising correlations with holistic scores assigned by human raters.

We plan to continue our research in the following directions: (1) we will improve the native speech norms for vowel durations, such as using the distribution of vowel durations rather than just the mean of durations in our feature computations; (2) we will investigate other aspects of pronunciation, e.g., consonant quality and word stress; (3) we will add other standard varieties of English (such as British, Canadian, Australian, etc) to the training corpus for the reference pronunciation model as the current model is trained on primarily North American English (NaE).

References

J. Bernstein. 1999. PhonePass testing: Structure and construct. Technical report, Ordinate Corporation.

C. Cucchiarini, H. Strik, and L. Boves. 1997a. Automatic evaluation of Dutch Pronunciation by using Speech Recognition Technology. In *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Santa Barbara, CA.

C. Cucchiarini, H. Strik, and L. Boves. 1997b. Using Speech Recognition Technology to Assess Foreign Speakers' Pronunciation of Dutch. In *3rd international symosium on the acquision of second language speech*, Klagenfurt, Austria.

J. Fiscus. 2009. Speech Recognition Scoring Toolkit (SCTK) Version 2.3.10.

H. Franco, V. Abrash, K. Precoda, H. Bratt, R. Rao, and J. Butzberger. 2000. The SRI EduSpeak system: Recognition and pronunciation scoring for language learning. In *InSTiLL (Intelligent Speech Technology in Language Learning)*, Dundee, Stotland.

D. Graff, J. Garofolo, J. Fiscus, W. Fisher, and D. Pallett. 1997. 1996 English Broadcast News Speech (HUB4).

C. Hacker, T. Cincarek, R. Grubn, S. Steidl, E. Noth, and H. Niemann. 2005. Pronunciation Feature Extraction. In *Proceedings of DAGM 2005*.

N. Moustroufas and V. Digalakis. 2007. Automatic pronunciation evaluation of foreign speakers using unknown text. *Computer Speech and Language*, 21(6):219–230.

L. Neumeyer, H. Franco, V. Digalakis, and M. Weintraub. 2000. Automatic Scoring of Pronunciation Quality. *Speech Communication*, 6.

S. M. Witt. 1999. *Use of Speech Recognition in Computer-assisted Language Learning*. Ph.D. thesis, University of Cambridge.

K. Zechner and I. Bejar. 2006. Towards Automatic Scoring of Non-Native Spontaneous Speech. In *NAACL-HLT*, NewYork NY.

K. Zechner, D. Higgins, and Xiaoming Xi. 2007. SpeechRater: A Construct-Driven Approach to Scoring Spontaneous Non-Native Speech. In *Proc. SLaTE*.

Performance Prediction for Exponential Language Models

Stanley F. Chen
IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598
stanchen@watson.ibm.com

Abstract

We investigate the task of performance prediction for language models belonging to the exponential family. First, we attempt to empirically discover a formula for predicting test set cross-entropy for n-gram language models. We build models over varying domains, data set sizes, and n-gram orders, and perform linear regression to see whether we can model test set performance as a simple function of training set performance and various model statistics. Remarkably, we find a simple relationship that predicts test set performance with a correlation of 0.9997. We analyze why this relationship holds and show that it holds for other exponential language models as well, including class-based models and minimum discrimination information models. Finally, we discuss how this relationship can be applied to improve language model performance.

1 Introduction

In this paper, we investigate the following question for language models belonging to the exponential family: given some training data and test data drawn from the same distribution, can we accurately predict the test set performance of a model estimated from the training data? This problem is known as *performance prediction* and is relevant for *model selection*, the task of selecting the best model from a set of candidate models given data.[1]

Let us first define some notation. Events have the form (x, y), where we attempt to predict the current word y given previous words x. We denote the training data as $\mathcal{D} = (x_1, y_1), \ldots, (x_D, y_D)$ and define $\tilde{p}(x, y) = \mathit{count}_{\mathcal{D}}(x, y)/D$ to be the empirical distribution of the training data. Similarly, we have a test set $\mathcal{D}^*$ and an associated empirical distribution $p^*(x, y)$. We take the performance of a conditional language model $p(y|x)$ to be the cross-entropy $\mathcal{H}(p^*, p)$ between the empirical test distribution p^* and the model $p(y|x)$:

$$\mathcal{H}(p^*, p) = -\sum_{x,y} p^*(x, y) \log p(y|x) \tag{1}$$

This is equivalent to the negative mean log-likelihood per event, as well as to log perplexity.

We only consider models in the exponential family. An exponential model $p_\Lambda(y|x)$ is a model with a set of *features* $\{f_1(x, y), \ldots, f_F(x, y)\}$ and equal number of parameters $\Lambda = \{\lambda_1, \ldots, \lambda_F\}$ where

$$p_\Lambda(y|x) = \frac{\exp(\sum_{i=1}^{F} \lambda_i f_i(x, y))}{Z_\Lambda(x)} \tag{2}$$

and where $Z_\Lambda(x)$ is a normalization factor.

One of the seminal methods for performance prediction is the Akaike Information Criterion (AIC) (Akaike, 1973). For a model, let $\hat{\Lambda}$ be the maximum likelihood estimate of Λ on some training data. Akaike derived the following estimate for the expected value of the test set cross-entropy $\mathcal{H}(p^*, p_{\hat{\Lambda}})$:

$$\mathcal{H}(p^*, p_{\hat{\Lambda}}) \approx \mathcal{H}(\tilde{p}, p_{\hat{\Lambda}}) + \frac{F}{D} \tag{3}$$

$\mathcal{H}(\tilde{p}, p_{\hat{\Lambda}})$ is the cross-entropy of the training set, F is the number of parameters in the model, and D is the number of events in the training data. However, maximum likelihood estimates for language models typically yield infinite cross-entropy on test data, and thus AIC behaves poorly for these domains.

In this work, instead of deriving a performance prediction relationship theoretically, we attempt to *empirically* discover a formula for predicting test performance. Initially, we consider only n-gram language models, and build models over varying domains, data set sizes, and n-gram orders. We perform linear regression to discover whether we can

[1] A long version of this paper can be found at (Chen, 2008).

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 450–458,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

model test set cross-entropy as a simple function of training set cross-entropy and other model statistics. For the 200+ n-gram models we evaluate, we find that the empirical relationship

$$\mathcal{H}(p^*, p_{\tilde{\Lambda}}) \approx \mathcal{H}(\tilde{p}, p_{\tilde{\Lambda}}) + \frac{\gamma}{D} \sum_{i=1}^{F} |\tilde{\lambda}_i| \qquad (4)$$

holds with a correlation of 0.9997 where γ is a constant and where $\tilde{\Lambda} = \{\tilde{\lambda}_i\}$ are *regularized* parameter estimates; *i.e.*, rather than estimating performance for maximum likelihood models as in AIC, we do this for regularized models. In other words, test set cross-entropy can be approximated by the sum of the training set cross-entropy and the scaled sum of the magnitudes of the model parameters.

To maximize the correlation achieved by eq. (4), we find that it is necessary to use the same regularization method and regularization hyperparameters across models and that the optimal value of γ depends on the values of the hyperparameters. Consequently, we first evaluate several types of regularization and find which of these (and which hyperparameter values) work best across all domains, and use these values in all subsequent experiments. While ℓ_2^2 regularization gives the best performance reported in the literature for n-gram models, we find here that $\ell_1 + \ell_2^2$ regularization works even better.

The organization of this paper is as follows: In Section 2, we evaluate various regularization techniques for n-gram models and select the method and hyperparameter values that give the best overall performance. In Section 3, we discuss experiments to find a formula for predicting n-gram model performance, and provide an explanation for why eq. (4) works so well. In Section 4, we evaluate how well eq. (4) holds for several class-based language models and minimum discrimination information models. Finally, in Sections 5 and 6 we discuss related work and conclusions.

2 Selecting Regularization Settings

In this section, we address the issue of how to perform regularization in our later experiments. Following the terminology of Dudík and Schapire (2006), the most widely-used and effective methods for regularizing exponential models are ℓ_1 regularization (Tibshirani, 1994; Kazama and Tsujii, 2003; Goodman, 2004) and ℓ_2^2 regularization (Lau, 1994; Chen and Rosenfeld, 2000; Lebanon and Lafferty, 2001). While not as popular, another regularization scheme that has been shown to be effective is *2-norm inequality* regularization (Kazama and Tsujii, 2003) which is an instance of $\ell_1 + \ell_2^2$ regularization as noted by Dudík and Schapire (2006). Under $\ell_1 + \ell_2^2$ regularization, the regularized parameter estimates $\tilde{\Lambda}$ are chosen to optimize the objective function

$$\mathcal{O}_{\ell_1+\ell_2^2}(\Lambda) = \mathcal{H}(\tilde{p}, p_\Lambda) + \frac{\alpha}{D} \sum_{i=1}^{F} |\lambda_i| + \frac{1}{2\sigma^2 D} \sum_{i=1}^{F} \lambda_i^2 \qquad (5)$$

Note that ℓ_1 regularization can be considered a special case of this (by taking $\sigma = \infty$) as can ℓ_2^2 regularization (by taking $\alpha = 0$).

Here, we evaluate ℓ_1, ℓ_2^2, and $\ell_1 + \ell_2^2$ regularization for exponential n-gram models. An exponential n-gram model contains a binary feature f_ω for each n'-gram ω occurring in the training data for $n' \leq n$, where $f_\omega(x, y) = 1$ iff xy ends in ω. We would like to find the regularization method and associated hyperparameters that work best across different domains, training set sizes, and n-gram orders. As it is computationally expensive to evaluate a large number of hyperparameter settings over a large collection of models, we divide this search into two phases. First, we evaluate a large set of hyperparameters on a limited set of models to come up with a short list of candidate hyperparameters. We then evaluate these candidates on our full model set to find the best one.

We build n-gram models over data from five different sources and consider three different vocabulary sizes for one source, giving us seven "domains"

	data source	token type	range of n	training sents.	voc. size
A	RH	letter	2–7	100–75k	27
B	WSJ	POS	2–7	100–30k	45
C	WSJ	word	2–5	100–100k	300
D	WSJ	word	2–5	100–100k	3k
E	WSJ	word	2–5	100–100k	21k
F	BN	word	2–5	100–100k	84k
G	SWB	word	2–5	100–100k	19k

Table 1: Statistics of data sets. RH = Random House dictionary; WSJ = Wall Street Journal; BN = Broadcast News; SWB = Switchboard.

in total. We refer to these domains by the letters *A*–*G*; summary statistics for each domain are given in Table 1. The domains *C*–*G* consist of regular word data, while domains *A* and *B* consist of letter and part-of-speech (POS) sequences, respectively. Domains *C*–*E* differ only in vocabulary.

For each domain, we first randomize the order of sentences in that data. We partition off two development sets and an evaluation set (5000 "sentences" each in domain *A* and 2500 sentences elsewhere) and use the remaining data as training data. In this way, we assure that our training and test data are drawn from the same distribution as is assumed in our later experiments. Training set sizes in sentences are 100, 300, 1000, 3000, etc., up to the maximums given in Table 1. Building models for each training set size and n-gram order in Table 1 gives us a total of 218 models over the seven domains.

In the first phase of hyperparameter search, we choose a subset of these models (57 total) and evaluate many different values for (α, σ^2) with $\ell_1+\ell_2^2$ regularization on each. We perform a grid search, trying each value $\alpha \in \{0.0, 0.1, 0.2, \ldots, 1.2\}$ with each value $\sigma^2 \in \{1, 1.2, 1.5, 2, 2.5, 3, 4, 5, 6, 7, 8, 10, \infty\}$ where $\sigma = \infty$ corresponds to ℓ_1 regularization and $\alpha = 0$ corresponds to ℓ_2^2 regularization. We use a variant of iterative scaling for parameter estimation. For each model and each (α, σ^2), we denote the cross-entropy of the development data as $H^m_{\alpha,\sigma}$ for the mth model, $m \in \{1, \ldots, 57\}$. Then, for each m and (α, σ^2), we can compute how much worse the settings (α, σ^2) perform with model m as compared to the best hyperparameter settings for that model:

$$\hat{H}^m_{\alpha,\sigma} = H^m_{\alpha,\sigma} - \min_{\alpha',\sigma'} H^m_{\alpha',\sigma'} \tag{6}$$

We would like to select (α, σ^2) for which $\hat{H}^m_{\alpha,\sigma}$ tends to be small; in particular, we choose (α, σ^2) that minimizes the root mean squared (RMS) error

$$\hat{H}^{\text{RMS}}_{\alpha,\sigma} = \sqrt{\frac{1}{57}\sum_{m=1}^{57}(\hat{H}^m_{\alpha,\sigma})^2} \tag{7}$$

For each of ℓ_1, ℓ_2^2, and $\ell_1 + \ell_2^2$ regularization, we retain the 6–8 best hyperparameter settings. To choose the best single hyperparameter setting from within this candidate set, we repeat the same analysis except over the full set of 218 models.

statistic	RMSE	coeff.
$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert$	0.043	0.938
$\frac{1}{D}\sum_{i:\tilde{\lambda}_i>0}\tilde{\lambda}_i$	0.044	0.939
$\frac{1}{D}\sum_{i=1}^{F}\tilde{\lambda}_i$	0.047	0.940
$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert^{\frac{4}{3}}$	0.162	0.755
$\frac{1}{D}\sum_{i=1}^{F}\lvert\tilde{\lambda}_i\rvert^{\frac{3}{2}}$	0.234	0.669
$\frac{1}{D}\sum_{i=1}^{F}\tilde{\lambda}_i^2$	0.429	0.443
$\frac{F_{\neq 0}}{D}$	0.709	1.289
$\frac{F_{\neq 0}\log D}{D}$	0.783	0.129
$\frac{F}{D}$	0.910	1.109
$\frac{F\log D}{D}$	0.952	0.112
1	1.487	1.698
$\frac{F}{D-F-1}$	2.232	-0.028
$\frac{F_{\neq 0}}{D-F_{\neq 0}-1}$	2.236	-0.023

Table 2: Root mean squared error (RMSE) in nats when predicting difference in development set and training set cross-entropy as linear function of a single statistic. The last column is the optimal coefficient for that statistic.

On the development sets, the (α, σ^2) value with the lowest squared error is (0.5, 6), and these are the hyperparameter settings we use in all later experiments unless otherwise noted. The RMS error, mean error, and maximum error for these hyperparameters on the evaluation sets are 0.011, 0.007, and 0.033 nats, respectively.[2] An error of 0.011 nats corresponds to a 1.1% difference in perplexity which is generally considered insignificant. Thus, we can achieve good performance across domains, data set sizes, and n-gram orders using a single set of hyperparameters as compared to optimizing hyperparameters separately for each model.

3 N-Gram Model Performance Prediction

Now that we have established which regularization method and hyperparameters to use, we attempt to empirically discover a simple formula for predicting the test set cross-entropy of regularized n-gram models. The basic strategy is as follows: We first build a large number of n-gram models over different domains, training set sizes, and n-gram orders. Then, we come up with a set of candidate statistics, *e.g.*, training set cross-entropy, number of features, etc., and do linear regression to try to best model test

[2] All cross-entropy values are reported in *nats*, or natural bits, equivalent to $\log_2 e$ regular bits. This will let us directly compare γ values with average discounts in Section 3.1.

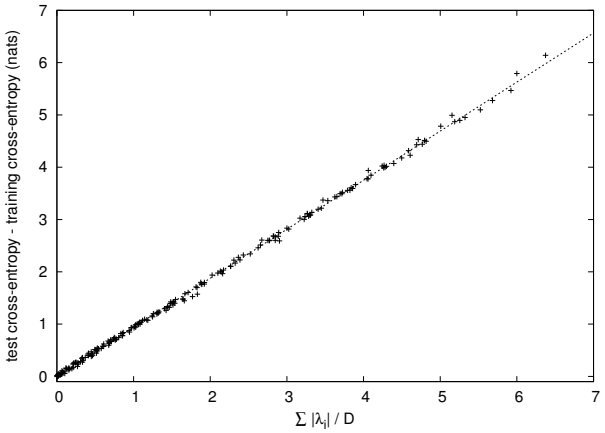


Figure 1: Graph of optimism on evaluation data *vs.* $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for various n-gram models under $\ell_1 + \ell_2^2$ regularization, $\alpha = 0.5$ and $\sigma^2 = 6$. The line represents the predicted optimism according to eq. (9) with $\gamma = 0.938$.

set cross-entropy as a linear function of these statistics. We assume that training and test data come from the same distribution; otherwise, it would be difficult to predict test performance.

We use the same 218 n-gram models as in Section 2. For each model, we compute training set cross-entropy $\mathcal{H}(\tilde{p}, p_{\tilde{\Lambda}})$ as well as all of the statistics listed on the left in Table 2. The statistics $\frac{F}{D}$, $\frac{F}{D-F-1}$, and $\frac{F \log D}{D}$ are motivated by AIC, AIC$_c$ (Hurvich and Tsai, 1989), and the Bayesian Information Criterion (Schwarz, 1978), respectively. As features f_i with $\tilde{\lambda}_i = 0$ have no effect, instead of F we also consider using $F_{\neq 0}$, the number of features f_i with $\tilde{\lambda}_i \neq 0$. The statistics $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ and $\frac{1}{D}\sum_{i=1}^{F}\tilde{\lambda}_i^2$ are motivated by eq. (5). The statistics with fractional exponents are suggested by Figure 2. The value 1 is present to handle constant offsets.

After some initial investigation, it became clear that training set cross-entropy is a very good (partial) predictor of test set cross-entropy with coefficient 1. As there is ample theoretical support for this, instead of fitting test set performance directly, we chose to model the difference between test and training performance as a function of the remaining statistics. This difference is sometimes referred to as the *optimism* of a model:

$$\textit{optimism}(p_{\tilde{\Lambda}}) \equiv \mathcal{H}(p^*, p_{\tilde{\Lambda}}) - \mathcal{H}(\tilde{p}, p_{\tilde{\Lambda}}) \qquad (8)$$

First, we attempt to model optimism as a linear function of a single statistic. For each statistic listed previously, we perform linear regression to minimize root mean squared error when predicting development set optimism. In Table 2, we display the RMSE and best coefficient for each statistic. We see that three statistics have by far the lowest error: $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$, $\frac{1}{D}\sum_{i:\tilde{\lambda}_i>0}\tilde{\lambda}_i$, and $\frac{1}{D}\sum_{i=1}^{F}\tilde{\lambda}_i$. In practice, most $\tilde{\lambda}_i$ in n-gram models are positive, so these statistics tend to have similar values. We choose the best ranked of these, $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$, and show in Section 3.1 why this statistic is more appealing than the others. Next, we investigate modeling optimism as a linear function of a *pair* of statistics. We find that the best RMSE for two variables (0.042) is only slightly lower than that for one (0.043), so it is doubtful that a second variable helps.

Thus, our analysis suggests that among our candidates, the best predictor of optimism is simply

$$\text{optimism} \approx \frac{\gamma}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i| \qquad (9)$$

where $\gamma = 0.938$, with this value being independent of domain, training set size, and n-gram order. In other words, the difference between test and training cross-entropy is a linear function of the sum of parameter magnitudes scaled per event. Substituting into eq. (8) and rearranging, we get eq. (4).

To assess the accuracy of eq. (4), we compute various statistics on our evaluation sets using the best γ from our development data, *i.e.*, $\gamma = 0.938$. In Figure 1, we graph optimism for the evaluation data against $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for each of our models; we see that the linear correlation is very good. The correlation between the actual and predicted cross-entropy on the evaluation data is 0.9997; the mean absolute prediction error is 0.030 nats; the RMSE is 0.043 nats; and the maximum absolute error is 0.166 nats. Thus, on average we can predict test performance to within 3% in perplexity, though in the worst case we may be off by as much as 18%.[3]

[3]The sampling variation in our test set selection limits the measured accuracy of our performance prediction. To give some idea of the size of this effect, we randomly selected 100 test sets in domain *D* of 2500 sentences each (as in our other experiments). We evaluated their cross-entropies using models trained on 100, 1k, 10k, and 100k sentences. The empiri-

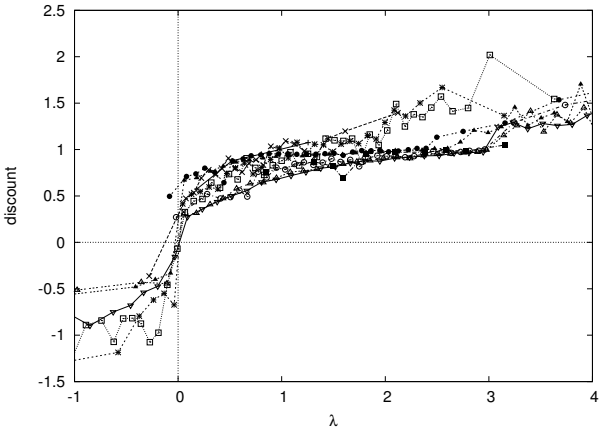


Figure 2: Smoothed graph of discount versus $\tilde{\lambda}_i$ for all features in ten different models built on domains *A* and *E*. Each smoothed point represents the average of at least 512 raw data points.

If we compute the prediction error of eq. (4) over the same models except using ℓ_1 or ℓ_2^2 regularization (with the best corresponding hyperparameter values found in Section 2), the prediction RMSE is 0.054 and 0.139 nats, respectively. Thus, we find that choosing hyperparameters carefully in Section 2 was important in doing well in performance prediction. While hyperparameters were chosen to optimize test performance rather than prediction accuracy, we find that the chosen hyperparameters are favorable for the latter task as well.

3.1 Why Does Prediction Work So Well?

The correlation in Figure 1 is remarkably high, and thus it begs for an explanation. First, let us express the difference in test and training cross-entropy for a model in terms of its parameters Λ. Substituting eq. (2) into eq. (1), we get

$$\mathcal{H}(p^*, p_\Lambda) = -\sum_{i=1}^{F} \lambda_i E_{p^*}[f_i] + \sum_x p^*(x) \log Z_\Lambda(x) \tag{10}$$

where $E_{p^*}[f_i] = \sum_{x,y} p^*(x,y) f_i(x,y)$. Then, we can express the difference in test and training performance as

$$\mathcal{H}(p^*, p_\Lambda) - \mathcal{H}(\tilde{p}, p_\Lambda) = \sum_{i=1}^{F} \lambda_i (E_{\tilde{p}}[f_i] - E_{p^*}[f_i]) + \sum_x (p^*(x) - \tilde{p}(x)) \log Z_\Lambda(x) \tag{11}$$

Ignoring the last term on the right, we see that optimism for exponential models is a linear function of the λ_i's with coefficients $E_{\tilde{p}}[f_i] - E_{p^*}[f_i]$.

Then, we can ask what $E_{\tilde{p}}[f_i] - E_{p^*}[f_i]$ values would let us satisfy eq. (4). Consider the relationship

$$(E_{\tilde{p}}[f_i] - E_{p^*}[f_i]) \times D \approx \gamma \operatorname{sgn} \tilde{\lambda}_i \tag{12}$$

If we substitute this into eq. (11) and ignore the last term on the right again, this gives us exactly eq. (4). We refer to the value $(E_{\tilde{p}}[f_i] - E_{p^*}[f_i]) \times D$ as the *discount* of a feature. It can be thought of as representing how many times less the feature occurs in the test data as opposed to the training data, if the test data were normalized to be the same size as the training data. Discounts for n-grams have been studied extensively, *e.g.*, (Good, 1953; Church and Gale, 1991; Chen and Goodman, 1998), and tend not to vary much across training set sizes.

We can check how well eq. (12) holds for actual regularized n-gram models. We construct a total of ten n-gram models on domains *A* and *E*. We build four letter 5-gram models on domain *A* on training sets ranging in size from 100 words to 30k words, and six models (either trigram or 5-gram) on domain *E* on training sets ranging from 100 sentences to 30k sentences. We create large development test sets (45k words for domain *A* and 70k sentences for domain *E*) to better estimate $E_{p^*}[f_i]$.

Because graphs of discounts as a function of $\tilde{\lambda}_i$ are very noisy, we smooth the data before plotting. We partition data points into buckets containing at least 512 points. We average all of the points in each bucket to get a "smoothed" data point, and plot this single point for each bucket. In Figure 2, we plot smoothed discounts as a function of $\tilde{\lambda}_i$ over the range $\tilde{\lambda}_i \in [-1, 4]$ for all ten models.

We see that eq. (12) holds at a very rough level over the $\tilde{\lambda}_i$ range displayed. If we examine how much different ranges of $\tilde{\lambda}_i$ contribute to the overall value of $\sum_{i=1}^{F} \tilde{\lambda}_i (E_{\tilde{p}}[f_i] - E_{p^*}[f_i])$, we find that the great majority of the mass (90–95%+) is concentrated in the range $\tilde{\lambda}_i \in [0, 4]$ for all ten models under consideration. Thus, to a first approximation, the reason that eq. (4) holds with $\gamma = 0.938$ is because on average, feature expectations have a discount of about this value for $\tilde{\lambda}_i$ in this range.[4]

cal standard deviation across test sets was found to be 0.0123, 0.0144, 0.0167, and 0.0174 nats, respectively. This effect can be mitigated by simply using larger test sets.

[4]This analysis provides some insight as to when eq. (4)

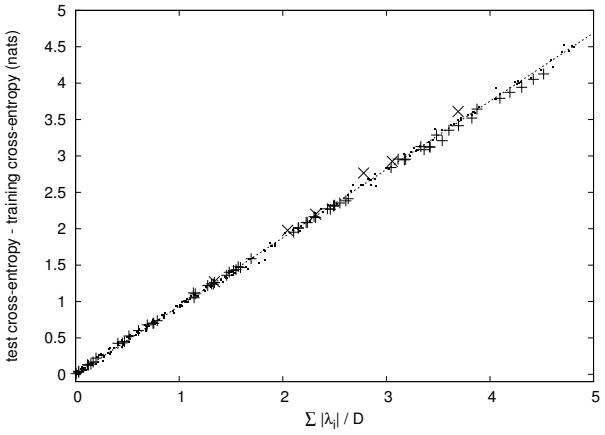


Figure 3: Graph of optimism on evaluation data *vs.* $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for various models. The '+' marks correspond to models **S**, **M**, and **L** over different training set sizes, n-gram orders, and numbers of classes. The '$\times$' marks correspond to MDI models over different n-gram orders and in-domain training set sizes. The line and small points are taken from Figure 1.

Due to space considerations, we only summarize our other findings; a longer discussion is provided in (Chen, 2008). We find that the absolute error in cross-entropy tends to be quite small across models for several reasons. For non-sparse models, there is significant variation in average discounts, but because $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ is low, the overall error is low. In contrast, sparse models are dominated by single-count n-grams with features whose average discount is quite close to $\gamma = 0.938$. Finally, the last term on the right in eq. (11) also plays a small but significant role in keeping the prediction error low.

4 Other Exponential Language Models

In (Chen, 2009), we show how eq. (4) can be used to motivate a novel class-based language model and a regularized version of minimum discrimination information (MDI) models (Della Pietra et al., 1992). In this section, we analyze whether in addition to word n-gram models, eq. (4) holds for these other exponential language models as well.

won't hold. For example, if a feature function f_i is doubled, its expectations and discount will also double. Thus, eq. (4) won't hold in general for models with continuous feature values, as average discounts may vary widely.

4.1 Class-Based Language Models

We assume a word w is always mapped to the same class $c(w)$. For a sentence $w_1 \cdots w_l$, we have

$$p(w_1 \cdots w_l) = \prod_{j=1}^{l+1} p(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) \times \prod_{j=1}^{l} p(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) \quad (13)$$

where $c_j = c(w_j)$ and where c_{l+1} is an end-of-sentence token. We use the notation $p_{\text{ng}}(y|\omega)$ to denote an exponential n-gram model as defined in Section 2, where we have features for each suffix of each ωy occurring in the training set. We use the notation $p_{\text{ng}}(y|\omega_1, \omega_2)$ to denote a model containing all features in the models $p_{\text{ng}}(y|\omega_1)$ and $p_{\text{ng}}(y|\omega_2)$.

We consider three class models, models **S**, **M**, and **L**, defined as

$$p_S(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) = p_{\text{ng}}(c_j|c_{j-2}c_{j-1})$$
$$p_S(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) = p_{\text{ng}}(w_j|c_j)$$

$$p_M(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) = p_{\text{ng}}(c_j|c_{j-2}c_{j-1}, w_{j-2}w_{j-1})$$
$$p_M(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) = p_{\text{ng}}(w_j|w_{j-2}w_{j-1}c_j)$$

$$p_L(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) = p_{\text{ng}}(c_j|w_{j-2}c_{j-2}w_{j-1}c_{j-1})$$
$$p_L(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) = p_{\text{ng}}(w_j|w_{j-2}c_{j-2}w_{j-1}c_{j-1}c_j)$$

Model **S** is an exponential version of the class-based n-gram model from (Brown et al., 1992); model **M** is a novel model introduced in (Chen, 2009); and model **L** is an exponential version of the model *ind-expredict* from (Goodman, 2001).

To evaluate whether eq. (4) can accurately predict test performance for these class-based models, we use the WSJ data and vocabulary from domain *E* and consider training set sizes of 1k, 10k, 100k, and 900k sentences. We create three different word classings containing 50, 150, and 500 classes using the algorithm of Brown et al. (1992) on the largest training set. For each training set and number of classes, we build both 3-gram and 4-gram versions of each of our three class models.

In Figure 3, we plot optimism (*i.e.*, test minus training cross-entropy) versus $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for these models (66 in total) on our WSJ evaluation set. The '+' marks correspond to our class n-gram models, while the small points replicate the points for word n-gram models from Figure 1. Remarkably, eq. (4) appears to accurately predict performance for our

class n-gram models using the same $\gamma = 0.938$ value found for word n-gram models. The mean absolute prediction error is 0.029 nats, comparable to that found for word n-gram models.

It is interesting that eq. (4) works for class-based models despite their being composed of two submodels, one for word prediction and one for class prediction. However, taking the log of eq. (13), we note that the cross-entropy of text can be expressed as the sum of the cross-entropy of its word tokens and the cross-entropy of its class tokens. It would not be surprising if eq. (4) holds separately for the class prediction model predicting class data and the word prediction model predicting word data, since all of these component models are basically n-gram models. Summing, this explains why eq. (4) holds for the whole class model.

4.2 Models with Prior Distributions

Minimum discrimination information models (Della Pietra et al., 1992) are exponential models with a *prior* distribution $q(y|x)$:

$$p_\Lambda(y|x) = q(y|x)\frac{\exp(\sum_{i=1}^{F}\lambda_i f_i(x,y))}{Z_\Lambda(x)} \qquad (14)$$

The central issue in performance prediction for MDI models is whether $q(y|x)$ needs to be accounted for. That is, if we assume q is an exponential model, should its parameters λ_i^q be included in the sum in eq. (4)? From eq. (11), we note that if $E_{\tilde{p}}[f_i] - E_{p^*}[f_i] = 0$ for a feature f_i, then the feature does not affect the difference between test and training cross-entropy (ignoring its impact on the last term). By assumption, the training and test set for p come from the same distribution while q is derived from an independent data set. It follows that we expect $E_{\tilde{p}}[f_i^q] - E_{p^*}[f_i^q]$ to be zero for features in q, and we should ignore q when applying eq. (4).

To evaluate whether eq. (4) holds for MDI models, we use the same WSJ training and evaluation sets from domain *E* as in Section 4.1. We consider three different training set sizes: 1k, 10k, and 100k sentences. To train q, we use the 100k sentence BN training set from domain *F*. We build both trigram and 4-gram versions of each model.

In Figure 3, we plot test minus training cross-entropy versus $\frac{1}{D}\sum_{i=1}^{F}|\tilde{\lambda}_i|$ for these models on our WSJ evaluation set; the '$\times$' marks correspond to the MDI models. As expected, eq. (4) appears to work quite well for MDI models using the same $\gamma = 0.938$ value as before; the mean absolute prediction error is 0.077 nats.

5 Related Work

We group existing performance prediction methods into two categories: *non-data-splitting* methods and *data-splitting* methods. In non-data-splitting methods, test performance is directly estimated from training set performance and/or other statistics of a model. Data-splitting methods involve partitioning training data into a truncated training set and a surrogate test set and using surrogate test set performance to estimate true performance.

The most popular non-data-splitting methods for predicting test set cross-entropy (or likelihood) are AIC and variants such as AIC$_c$, quasi-AIC (QAIC), and QAIC$_c$ (Akaike, 1973; Hurvich and Tsai, 1989; Lebreton et al., 1992). In Section 3, we considered performance prediction formulae with the same form as AIC and AIC$_c$ (except using regularized parameter estimates), and neither performed as well as eq. (4); *e.g.*, see Table 2.

There are many techniques for bounding test set classification error including the Occam's Razor bound (Blumer et al., 1987; McAllester, 1999), PAC-Bayes bound (McAllester, 1999), and the sample compression bound (Littlestone and Warmuth, 1986; Floyd and Warmuth, 1995). These methods derive theoretical guarantees that the true error rate of a classifier will be below (or above) some value with a certain probability. Langford (2005) evaluates these techniques over many data sets; while the bounds can sometimes be fairly tight, in many data sets the bounds are quite loose.

When learning an element from a set of target classifiers, the Vapnik-Chervonenkis (VC) dimension of the set can be used to bound the true error rate relative to the training error rate with some probability (Vapnik, 1998); this technique has been used to compute error bounds for many types of classifiers. For example, Bartlett (1998) shows that for a neural network with small weights and small training set squared error, the true error depends on the size of its weights rather than the number of weights; this finding is similar in spirit to eq. (4).

In practice, the most accurate methods for performance prediction in many contexts are data-splitting methods (Guyon et al., 2006). These techniques include the hold-out method; leave-one-out and k-fold cross-validation; and bootstrapping (Allen, 1974; Stone, 1974; Geisser, 1975; Craven and Wahba, 1979; Efron, 1983). However, unlike non-data-splitting methods, these methods do not lend themselves well to providing insight into model design as discussed in Section 6.

6 Discussion

We show that for several types of exponential language models, it is possible to accurately predict the cross-entropy of test data using the simple relationship given in eq. (4). When using $\ell_1 + \ell_2^2$ regularization with $(\alpha = 0.5, \sigma^2 = 6)$, the value $\gamma = 0.938$ works well across varying model types, domains, vocabulary sizes, training set sizes, and n-gram orders, yielding a mean absolute error of about 0.03 nats (3% in perplexity). We evaluate $\sim$300 language models in total, including word and class n-gram models and n-gram models with prior distributions.

While there has been a great deal of work in performance prediction, the vast majority of work on non-data-splitting methods has focused on finding theoretically-motivated approximations or probabilistic bounds on test performance. In contrast, we developed eq. (4) on a purely empirical basis, and there has been little, if any, existing work that has shown comparable performance prediction accuracy over such a large number of models and data sets. In addition, there has been little, if any, previous work on performance prediction for language modeling.[5]

While eq. (4) performs well as compared to other non-data-splitting methods for performance prediction, the prediction error can be several percent in perplexity, which means we cannot reliably rank models that are close in quality. In addition, in speech recognition and many other applications, an external test set is typically provided, which means we can measure test set performance directly. Thus, in practice, eq. (4) is not terribly useful for the task of model selection; instead, what eq. (4) gives us is insight into *model design*. That is, instead of selecting between candidate models *once they have been built* as in model selection, it is desirable to be able to select between models at the *model design* stage. Being able to intelligently compare models (without implementation) requires that we know which aspects of a model impact test performance, and this is exactly what eq. (4) tells us.

Intuitively, simpler models should perform better on test data given equivalent training performance, and model structure (as opposed to parameter values) is an important aspect of the complexity of a model. Accordingly, there have been many methods for model selection that measure the size of a model in terms of the number of features or parameters in the model, *e.g.*, (Akaike, 1973; Rissanen, 1978; Schwarz, 1978). Surprisingly, for exponential language models, the number of model parameters seems to matter not at all; all that matters are the magnitudes of the parameter values. Consequently, one can improve such models by adding features (or a prior model) that reduce parameter values while maintaining training performance.

In (Chen, 2009), we show how these ideas can be used to motivate heuristics for improving the performance of existing language models, and use these heuristics to develop a novel class-based model and a regularized version of MDI models that outperform comparable methods in both perplexity and speech recognition word-error rate on WSJ data. In addition, we show how the tradeoff between training set performance and model size impacts aspects of language modeling as diverse as backoff n-gram features, class-based models, and domain adaptation. In sum, eq. (4) provides a new and valuable framework for characterizing, analyzing, and designing statistical models.

Acknowledgements

We thank Bhuvana Ramabhadran and the anonymous reviewers for their comments on this and earlier versions of the paper.

References

Hirotugu Akaike. 1973. Information theory and an extension of the maximum likelihood principle. In *Sec-*

[5] Here, we refer to predicting test set performance from *training set* performance and other model statistics. However, there has been a good deal of work on predicting speech recognition word-error rate from *test set* perplexity and other statistics, *e.g.*, (Klakow and Peters, 2002).

ond Intl. Symp. on Information Theory, pp. 267–281.

David M. Allen. 1974. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16(1):125–127.

Peter L. Bartlett. 1998. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Trans. on Information Theory*, 44(2):525–536.

Alselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. 1987. Occam's razor. *Information Processing Letters*, 24(6):377–380.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Tech. Report TR-10-98, Harvard Univ.

Stanley F. Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for maximum entropy models. *IEEE Trans. Speech and Aud. Proc.*, 8(1):37–50.

Stanley F. Chen. 2008. Performance prediction for exponential language models. Tech. Report RC 24671, IBM Research Division, October.

Stanley F. Chen. 2009. Shrinking exponential language models. In *Proc. of HLT-NAACL*.

Kenneth W. Church and William A. Gale. 1991. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5:19–54.

Peter Craven and Grace Wahba. 1979. Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31:377–403.

Stephen Della Pietra, Vincent Della Pietra, Robert L. Mercer, and Salim Roukos. 1992. Adaptive language modeling using minimum discriminant estimation. In *Proc. Speech and Natural Lang. DARPA Workshop*.

Miroslav Dudík and Robert E. Schapire. 2006. Maximum entropy distribution estimation with generalized regularization. In *Proc. of COLT*.

Bradley Efron. 1983. Estimating the error rate of a prediction rule: Improvement on cross-validation. *J. of the American Statistical Assoc.*, 78(382):316–331.

Sally Floyd and Manfred Warmuth. 1995. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304.

Seymour Geisser. 1975. The predictive sample reuse method with applications. *J. of the American Statistical Assoc.*, 70:320–328.

I.J. Good. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3 and 4):237–264.

Joshua T. Goodman. 2001. A bit of progress in language modeling. MSR-TR-2001-72, Microsoft Research.

Joshua Goodman. 2004. Exponential priors for maximum entropy models. In *Proc. of NAACL*.

Isabelle Guyon, Amir Saffari, Gideon Dror, and Joachim Buhmann. 2006. Performance prediction challenge. In *Proc. of Intl. Conference on Neural Networks (IJCNN06), IEEE World Congress on Computational Intelligence (WCCI06)*, pp. 2958–2965, July.

Clifford M. Hurvich and Chih-Ling Tsai. 1989. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, June.

Jun'ichi Kazama and Jun'ichi Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proc. of EMNLP*, pp. 137–144.

Dietrich Klakow and Jochen Peters. 2002. Testing the correlation of word error rate and perplexity. *Speech Communications*, 38(1):19–28.

John Langford. 2005. Tutorial on practical prediction theory for classification. *J. of Machine Learning Research*, 6:273–306.

Raymond Lau. 1994. Adaptive statistical language modelling. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Guy Lebanon and John Lafferty. 2001. Boosting and maximum likelihood for exponential models. Tech. Report CMU-CS-01-144, Carnegie Mellon Univ.

Jean-Dominique Lebreton, Kenneth P. Burnham, Jean Clobert, and David R. Anderson. 1992. Modeling survival and testing biological hypotheses using marked animals: a unified approach with case studies. *Ecological Monographs*, 62:67–118.

Nick Littlestone and Manfred K. Warmuth. 1986. Relating data compression and learnability. Tech. report, Univ. of California, Santa Cruz.

David A. McAllester. 1999. PAC-Bayesian model averaging. In *Proc. of COLT*, pp. 164–170.

Jorma Rissanen. 1978. Modeling by the shortest data description. *Automatica*, 14:465–471.

Gideon Schwarz. 1978. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464.

M. Stone. 1974. Cross-validatory choice and assessment of statistical predictions. *J. of the Royal Statistical Society B*, 36:111–147.

Robert Tibshirani. 1994. Regression shrinkage and selection via the lasso. Tech. report, Univ. of Toronto.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley, New York.

Tied–Mixture Language Modeling in Continuous Space

Ruhi Sarikaya
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
sarikaya@us.ibm.com

Mohamed Afify
Orange Labs.
Cairo, Egypt
mohamed_afify2001@yahoo.com

Brian Kingsbury
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
bedk@us.ibm.com

Abstract

This paper presents a new perspective to the language modeling problem by moving the word representations and modeling into the continuous space. In a previous work we introduced Gaussian-Mixture Language Model (GMLM) and presented some initial experiments. Here, we propose Tied-Mixture Language Model (TMLM), which does not have the model parameter estimation problems that GMLM has. TMLM provides a great deal of parameter tying across words, hence achieves robust parameter estimation. As such, TMLM can estimate the probability of any word that has as few as two occurrences in the training data. The speech recognition experiments with the TMLM show improvement over the word trigram model.

1 Introduction

Despite numerous studies demonstrating the serious short-comings of the n–gram language models, it has been surprisingly difficult to outperform n–gram language models consistently across different domains, tasks and languages. It is well-known that n–gram language models are not effective in modeling long range lexical, syntactic and semantic dependencies. Nevertheless, n–gram models have been very appealing due to their simplicity; they require only a plain corpus of data to train the model. The improvements obtained by some more elaborate language models (Chelba & Jelinek, 2000; Erdogan et al., 2005) come from the explicit use of syntactic and semantic knowledge put into the annotated corpus.

In addition to the mentioned problems above, traditional n–gram language models do not lend themselves easily to rapid and effective adaptation and discriminative training. A typical n–gram model contains millions of parameters and has no structure capturing dependencies and relationships between the words beyond a limited local context. These parameters are estimated from the empirical distributions, and suffer from data sparseness. n–gram language model adaptation (to new domain, speaker, genre and language) is difficult, simply because of the large number of parameters, for which large amount of adaptation data is required. Instead of updating model parameters with an adaptation method, the typical practice is to collect some data in the target domain and build a domain specific language model. The domain specific language model is interpolated with a generic language model trained on a larger domain independent data to achieve robustness. On the other hand, rapid adaptation for acoustic modeling, using such methods as Maximum Likelihood Linear Regression (MLLR) (Legetter & Woodland, 1995), is possible using very small amount of acoustic data, thanks to the inherent structure of acoustic models that allow large degrees of parameter tying across different words (several thousand context dependent states are shared by all the words in the dictionary). Likewise, even though discriminatively trained acoustic models have been widely used, discriminatively trained languages models (Roark et al., 2007) have not widely accepted as a standard practice yet.

In this study, we present a new perspective to the language modeling. In this perspective, words are not treated as discrete entities but rather vectors of real numbers. As a result, long–term semantic relationships between the words could be quantified and can be integrated into a model. The proposed formulation casts the language modeling problem as

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 459–467,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

an acoustic modeling problem in speech recognition. This approach opens up new possibilities from rapid and effective adaptation of language models to using discriminative acoustic modeling tools and methods, such as Minimum Phone Error (MPE) (Povey & Woodland, 2002) training to train discriminative language models.

We introduced the idea of language modeling in continuous space from the acoustic modeling perspective and proposed Gaussian Mixture Language Model (GMLM) (Afify et al., 2007). However, GMLM has model parameter estimation problems. In GMLM each word is represented by a specific set of Gaussian mixtures. Robust parameter estimation of the Gaussian mixtures requires hundreds or even thousands of examples. As a result, we were able to estimate the GMLM probabilities only for words that have at least 50 or more examples. Essentially, this was meant to estimate the GMLM probabilities for only about top 10% of the words in the vocabulary. Not surprisingly, we have not observed improvements in speech recognition accuracy (Afify et al., 2007). Tied-Mixture Language Model (TMLM) does not have these requirements in model estimation. In fact, language model probabilities can be estimated for words having as few as two occurrences in the training data.

The concept of language modeling in continuous space was previously proposed (Bengio et al., 2003; Schwenk & Gauvain, 2003) using Neural Networks. However, our method offers several potential advantages over (Schwenk & Gauvain, 2003) including adaptation, and modeling of semantic dependencies because of the way we represent the words in the continuous space. Moreover, our method also allows efficient model training using large amounts of training data, thanks to the acoustic modeling tools and methods which are optimized to handle large amounts of data efficiently.

It is important to note that we have to realize the full potential of the proposed model, before investigating the potential benefits such as adaptation and discriminative training. To this end, we propose TMLM, which does not have the problems GMLM has and, unlike GMLM we report improvements in speech recognition over the corresponding n–gram models.

The rest of the paper is organized as follows. Section 2 presents the concept of language modeling in continuous space. Section 3 describes the tied–mixture modeling. Speech recognition architecture is summarized in Section 4, followed by the experimental results in Section 5. Section 6 discusses various issues with the proposed method and finally, Section 7 summarizes our findings.

2 Language Modeling In Continuous Space

The language model training in continuous space has three main steps; namely, creation of a co–occurrence matrix, mapping discrete words into a continuous parameter space in the form of vectors of real numbers and training a statistical parametric model. Now, we will describe each step in detail.

2.1 Creation of a co–occurrence Matrix

There are many ways that discrete words can be mapped into a continuous space. The approach we take is based on Latent Semantic Analysis (LSA) (Deerwester et al., 1990), and begins with the creation of a co–occurrence matrix. The co–occurrence matrix can be constructed in several ways, depending on the morphological complexity of the language. For a morphologically impoverished language, such as English the co–occurrence matrix can be constructed using word bigram co–occurrences. For morphologically rich languages, there are several options to construct a co–occurrence matrix. For example, the co–occurrence matrix can be constructed using either words (word–word co–occurrences) or morphemes (morpheme–morpheme co–occurrences), which are obtained after morphologically tokenizing the entire corpus. In addition to word–word or morpheme–morpheme co–occurrence matrices, a word–morpheme co–occurrence matrix can also be constructed. A word w can be decomposed into a set of prefixes, stem and suffixes: $w = [pfx_1 + pfx_2 + pfx_n + stem + sfx_1 + sfx_2 + sfx_n]$. The columns of such a matrix contain words and the rows contain the corresponding morphological decomposition (i.e. morphemes) making up the word. The decomposition of this matrix (as will be described in the next sub-section) can allow joint modeling of words and morphemes in one model.

In this study, we use morpheme level bigram co–occurrences to construct the matrix. All the morpheme[1] bigrams are accumulated for the entire corpus to fill in the entries of a co–occurrence matrix, C, where $C(w_i, w_j)$ denotes the counts for which word w_i follows word w_j in the corpus. This is a large, but very sparse matrix, since typically a small number of words follow a given word. Because of its large size and sparsity, Singular Value Decomposition (SVD) is a natural choice for producing a reduced-rank approximation of this matrix.

The co–occurrence matrices typically contain a small number of high frequency events and a large number of less frequent events. Since SVD derives a compact approximation of the co–occurrence matrix that is optimal in the least–square sense, it best models these high-frequency events, which may not be the most informative. Therefore, the entries of a word-pair co–occurrence matrix are smoothed according to the following expression:

$$\hat{C}(w_i, w_j) = \log(C(w_i, w_j) + 1) \tag{1}$$

Following the notation presented in (Bellegarda, 2000) we proceed to perform the SVD as follows:

$$\hat{C} \approx USV^T \tag{2}$$

where U is a left singular matrix with row vectors u_i $(1 \leq i \leq M)$ and dimension $M \times R$. S is a diagonal matrix of singular values with dimension $R \times R$. V is a right singular matrix with row vectors v_j $(1 \leq j \leq N)$ and dimension $N \times R$. R is the order of the decomposition and $R \ll min(M, N)$. M and N are the vocabulary sizes on the rows and columns of the co–occurrence matrix, respectively. For word–word or morpheme–morpheme co–occurrence matrices $M = N$, but for word–morpheme co–occurrence matrix, M is the number of unique words in the training corpus and N is the number of unique morphemes in morphologically tokenized training corpus.

2.2 Mapping Words into Continuous Space

The continuous space for the words listed on the rows of the co–occurrence matrix is defined as the space spanned by the column vectors of $A_{M \times R} = US$. Similarly, the continuous space for the words on the columns are defined as the space spanned by the row vectors of $B_{R \times N} = SV^T$. Here, for a word–word co–occurrence matrix, each of the scaled vectors (by S) in the columns of A and rows of B are called latent word history vectors for the forward and backward bigrams, respectively. Now, a bigram $w_{ij} = (w_i, w_j)$ $(1 \leq i, j \leq M)$ is represented as a vector of dimension $M \times 1$, where the i^{th} entry of w_{ij} is 1 and the remaining ones are zero. This vector is mapped to a lower dimensional vector $\hat{w}_{ij}$ by:

$$\hat{w}_{ij} = A^T w_{ij} \tag{3}$$

where $\hat{w}_{ij}$ has dimension of $R \times 1$. Similarly, the backward bigram w_{ji} $(1 \leq j, i \leq N)$ is mapped to a lower dimensional vector $\hat{w}_{ji}$ by:

$$\hat{w}_{ji} = B w_{ji} \tag{4}$$

where $\hat{w}_{ji}$ has dimension of $R \times 1$. Note that for a word–morpheme co–occurrence matrix the rows of B would contain latent morpheme vectors.

Since a trigram history consists of two bigram histories, a trigram history vector is obtained by concatenating two bigram vectors. Having generated the features, now we explain the structure of the parametric model and how to train it for language modeling in continuous space.

2.3 Parametric Model Training in Continuous Space

Recalling the necessary inputs to train an acoustic model for speech recognition would be helpful to explain the new language modeling method. The acoustic model training in speech recognition needs three inputs: 1) features (extracted from the speech waveform), 2) transcriptions of the speech waveforms and 3) baseforms, which show the pronunciation of each word in the vocabulary. We propose to model the language model using HMMs. The HMM parameters are estimated in such way that the given set of observations is represented by the model in the "best" way. The "best" can be defined in various ways. One obvious choice is to use Maximum Likelihood (ML) criterion. In ML, we maximize the probability of a given sequence of observations O, belonging to a given class, given the HMM λ of the class, with respect to the parameters of the model λ.

[1]For the generality of the notation, from now on we use "word" instead of "morpheme".

This probability is the total likelihood of the observations and can be expressed mathematically as:

$$L_{tot} = p(O|\lambda) \tag{5}$$

However, there is no known way to analytically solve for the model $\lambda = \{A, B, \pi\}$, which maximize the quantity L_{tot}, where A is the transition probabilities, B is the observation probabilities, and π is the initial state distribution. But we can choose model parameters such that it is locally maximized, using an iterative procedure, like Baum-Welch method (Baum et al., 1970).

The objective function given in Eq. 5 is the same objective function used to estimate the parameters of an HMM based acoustic model. By drawing an analogy between the acoustic model training and language modeling in continuous space, the history vectors are considered as the acoustic observations (feature vectors) and the next word to be predicted is considered as the label the acoustic features belong to, and words with their morphological decompositions can be considered as the lexicon or dictionary. Fig. 1 presents the topology of the model for modeling a word sequence of 3 words. Each word is modeled with a single state left–to–right HMM topology. Using a morphologically rich language (or a character based language like Chinese) to explain how HMMs can be used for language modeling will be helpful. In the figure, let the states be the words and the observations that they emit are the morphemes (or characters in the case of Chinese). The same topology (3 states) can also be used to model a single word, where the first state models the prefixes, the middle state models the stem and the final state models the suffixes. In this case, words are represented by network of morphemes. Each path in a word network represents a segmentation (or "pronunciation") of the word.

The basic idea of the proposed modeling is to create a separate model for each word of the language and use the language model corpus to estimate the parameters of the model. However, one could argue that the basic model could be improved by taking the contexts of the morphemes into account. Instead of building a single HMM for each word, several models could be trained according to the context of the morphemes. These models are called context–dependent morphemes. The most obvious choice is to use both left and right neighbor of a morpheme as context, and creating, what we call tri–morphemes. In principal even if context-dependent morphemes could improve the modeling accuracy, the number of models increase substantially. For a vocabulary size of V, the number of tri–morpheme could be as high as V^3. However, most of the tri–morphemes are either rare or will not be observed in the training data altogether.

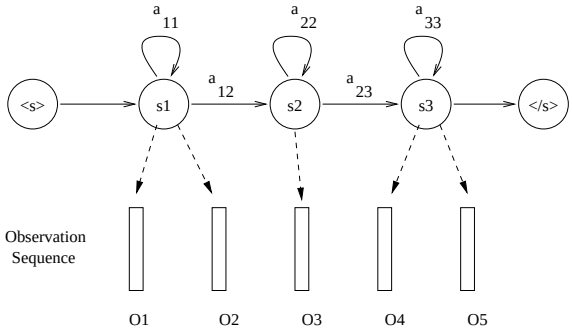


Figure 1: HMM topology for language modeling in continuous space.

Decision tree is one approach that can solve this problem. The main idea is to find similar tri–morphemes and share the parameters between them. The decision tree uses a top-down approach to split the samples, which are in a single cluster at the root of the tree, into smaller clusters by asking questions about the current morpheme and its context. In our case, the questions could be syntactic and/or semantic in nature.

What we hope for is that in the new continuous space there is some form of distance or similarity between histories such that histories not observed in the data for some words are smoothed by similar observed histories.

2.4 Summary of the Continuous Language Model Training and Using it for Decoding

In the upper part of Fig. 2 the language model training steps are shown. The training process starts with the language model training corpus. From the sentences a bigram word co–occurrence matrix is constructed. This is a square matrix where the number of rows (columns) equal to the vocabulary size of the training data. The bigram co–occurrence ma-

trix is decomposed using SVD. The columns of the left–singular matrix obtained from SVD is used to map the bigram word histories into a lower dimensional continuous parameter space. The projected word history vectors are stacked together depending on the size of the n–gram. For example, for trigram modeling two history vectors are stacked together. Even though, we have not done so, at this stage one could cluster the word histories for robust parameter estimation. Now, the feature vectors, their corresponding transcriptions and the lexicon (baseforms) are ready to perform the "acoustic model training". One could use maximum likelihood criterion or any other objective function such as Minimum Phone Error (MPE) training to estimate the language model parameters in the continuous space.

The decoding phase could employ an adaptation step, if one wants to adapt the language model to a different domain, speaker or genre. Then, given a hypothesized sequence of words the decoder extracts the corresponding feature vectors. The feature vectors are used to estimate the likelihood of the word sequence using the HMM parameters. This likelihood is used to compute the probability of the word sequence. Next, we introduce Tied–Mixture Modeling, which is a special HMM structure to robustly estimate model parameters.

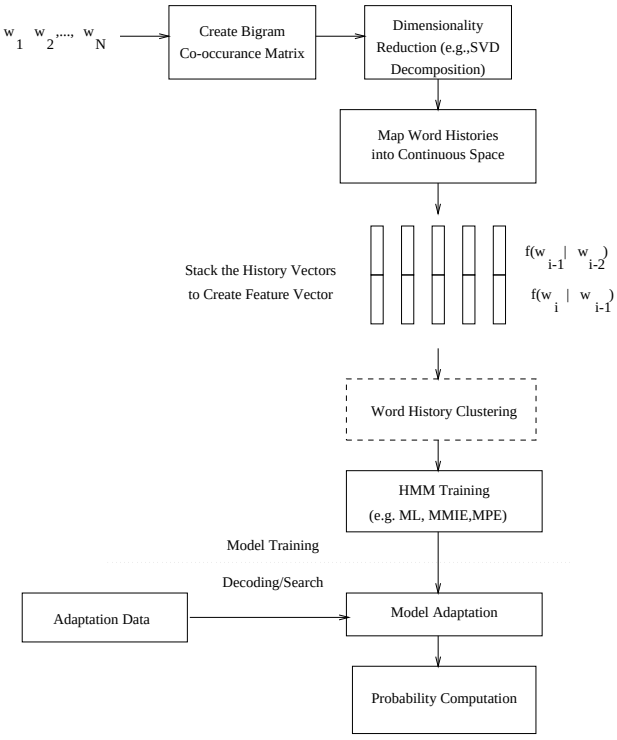


Figure 2: Language Model Training and Adaptation in Continuous Space.

3 Tied–Mixture Modeling

Hidden Markov Models (HMMs) have been extensively used virtually in all aspects of speech and language processing. In speech recognition area continuous-density HMMs have been the standard for modeling speech signals, where several thousand context–dependent states have their own Gaussian density functions to model different speech sounds. Typically, speech data have hundreds of millions of frames, which are sufficient to robustly estimate the model parameters. The amount of data for language modeling is orders of magnitude less than that of the acoustic data in continuous space. Tied–Mixture Hidden Markov Models (TM–HMMs) (Bellegarda & Nahamoo, 1989; Huang & Jack, 1988) have a better decoupling between the number of Gaussians and the number of states compared to continuous density HMMs. The TM–HMM is useful for language modeling because it allows us to choose the number of Gaussian densities and the number of mixture weights independently. Much more data is required to reliably estimate Gaussian densities than to estimate mixture weights.

The evaluation of the observation density functions for TM–HMMs can be time consuming due to the large mixture weight vector and due to the fact that for each frame all Gaussians have to be evaluated. However, there are a number of solutions proposed in the past that significantly reduces the computation (Duchateau et al., 1998).

The function $p(w \mid h)$, defined in a continuous space, represents the conditional probability of the word w given the history h. In general, h contains previous words and additional information (e.g. part-of-speech (POS) tags for the previous words) that may help to the prediction of the next word. Unlike TM–HMMs, using a separate HMM for each word as in the case of Gaussian Mixture Models (GMMs), to represent the probability distribution functions results in the estimation problems for the model parameters since each n–gram does not have hundreds of examples. TM–HMMs use Gaussian mixture probability density functions per

state in which a single set of Gaussians is shared among all states:

$$p(o|w) = \sum_{j}^{J} c_{w,j} \mathcal{N}_j(o, \mu_{w,j}, \Sigma_{w,j}) \qquad (6)$$

where w is the state, $\mathcal{N}_j$ is the jth Gaussian, and o is the observation (i.e. history) vectors. and J is the number of component mixtures in the TM-HMM. In order to avoid zero variance in word mapping into continuous space, all the latent word vectors are added a small amount of white noise.

The TM–HMM topology is given in Fig. 3. Each state models a word and they all share the same set of Gaussian densities. However, each state has a specific set of mixture weights associated with them. This topology can model a word–sequence that consist of three words in them. The TM–HMM estimates the probability of observing the history vectors (h) for a given word w. However, what we need is the posterior probability $p(w \mid h)$ of observing w as the next word given the history, h. This can be obtained using the Bayes rule:

$$p(w|h) = \frac{p(h|w)p(w)}{p(h)} \qquad (7)$$

$$= \frac{p(h|w)p(w)}{\sum_{v=1}^{V} p(h|v)p(v)} \qquad (8)$$

where $p(w)$ is the unigram probability of the word w. The unigram probabilities can also be substituted for more accurate higher order n–gram probabilities. If this n–gram has an order that is equal to or greater than the one used in defining the continuous contexts h, then the TMLM can be viewed as performing a kind of smoothing of the original n–gram model:

$$P_s(w \mid h) = \frac{P(w \mid h)p(h \mid w)}{\sum_{v=1}^{V} P(v \mid h)p(h \mid v)} \qquad (9)$$

where $P_s(w \mid h)$ and $P(w \mid h)$ are the smoothed and original n–grams.

The TM–HMM parameters are estimated through an iterative procedure called the Baum-Welch, or forward-backward, algorithm (Baum et al., 1970). The algorithm locally maximizes the likelihood function via an iterative procedure. This type of training is identical to training continuous density HMMs except the Gaussians are tied across all arcs. For the model estimation equations the readers are referred to (Bellegarda & Nahamoo, 1989; Huang & Jack, 1988).

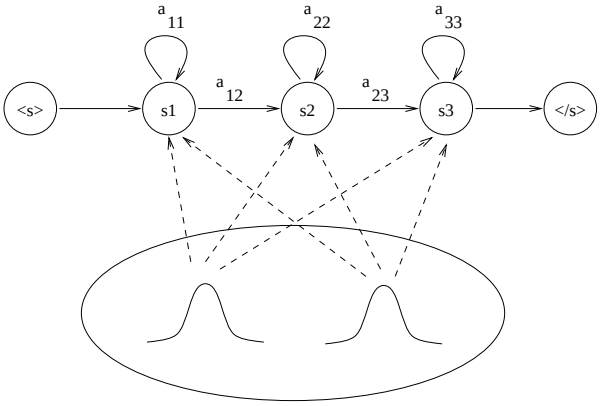


Figure 3: Tied-Mixture HMM topology for language modeling in continuous space. The mixtures are tied across states. Each state represents a word. The TM-HMM is completely defined with the mixture weights, mixture densities and transition probabilities.

Next, we introduce the speech recognition system used for the experiments.

4 Speech Recognition Architecture

The speech recognition experiments are carried out on the Iraqi Arabic side of an English to Iraqi Arabic speech-to-speech translation task. This task covers the military and medical domains. The acoustic data has about 200 hours of conversational speech collected in the context of a DARPA supported speech-to-speech (S2S) translation project (Gao et al., 2006).

The feature vectors for training acoustic models are generated as follows. The speech data is sampled at 16kHz and the feature vectors are computed every 10ms. First, 24-dimensional MFCC features are extracted and appended with the frame energy. The feature vector is then mean and energy normalized. Nine vectors, including the current vector and four vectors from its right and left contexts, are stacked leading to a 216-dimensional parameter space. The feature space is finally reduced from 216 to 40 dimensions using a combination of linear discriminant analysis (LDA), feature space MLLR (fMLLR) and feature space MPE (fMPE) training (Povey et al.,

2005). The baseline speech recognition system used in our experiments is the state–of–the–art and produces a competitive performance.

The phone set consists of 33 graphemes representing speech and silence for acoustic modeling. These graphemes correspond to letters in Arabic plus silence and short pause models. Short vowels are implicitly modeled in the neighboring graphemes. Feature vectors are first aligned, using initial models, to model states. A decision tree is then built for each state using the aligned feature vectors by asking questions about the phonetic context; quinphone questions are used in this case. The resulting tree has about 3K leaves. Each leaf is then modeled using a Gaussian mixture model. These models are first bootstrapped and then refined using three iterations of forward–backward training. The current system has about 75K Gaussians.

The language model training data has 2.8M words with 98K unique words and it includes acoustic model training data as a subset. The morphologically analyzed training data has 58K unique vocabulary items. The pronunciation lexicon consists of the grapheme mappings of these unique words. The mapping to graphemes is one-to-one and there are very few pronunciation variants that are supplied manually mainly for numbers. A statistical trigram language model using Modified Kneser-Ney smoothing (Chen& Goodman, 1996) has been built using the training data, which is referred to as Word-3gr.

For decoding a static decoding graph is compiled by composing the language model, the pronunciation lexicon, the decision tree, and the HMM graphs. This static decoding scheme, which compiles the recognition network off–line before decoding, is widely adopted in speech recognition (Riley et al., 2002). The resulting graph is further optimized using determinization and minimization to achieve a relatively compact structure. Decoding is performed on this graph using a Viterbi beam search.

5 Experimental Results

We used the following TMLM parameters to build the model. The SVD projection size is set to 200 (i.e. $R = 200$) for each bigram history. This results into a trigram history vector of size 400. This vector is further projected down to a 50 dimensional feature space using LDA transform. The total number of Gaussian densities used for the TM–HMM is set to 1024. In order to find the overall relationship between trigram and TMLM probabilities we show the scatter plot of the trigram and TMMT probabilities in Fig. 4. While calculating the TMLM score the TMLM likelihood generated by the model is divided by 40 to balance its dynamic range with that of the n–gram model. Most of the probabilities lie along the diagonal line. However, some trigram probabilities are modulated making TMLM probabilities quite different than the corresponding trigram probabilities. Analysis of TMLM probabilities with respect to the trigram probabilities would be an interesting future research.

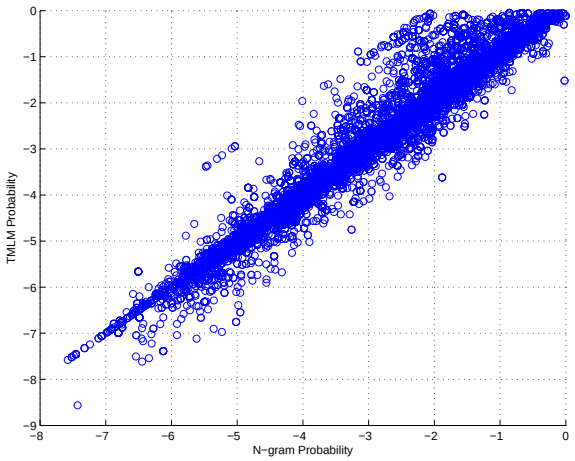


Figure 4: Scatter plot of the n–gram and TMLM probabilities.

We conducted the speech recognition language modeling experiments on 3 testsets: TestA, TestB and TestC. All three test sets are from July'07 official evaluations of the IBM's speech-to-speech translation system by DARPA. TestA consists of sentences spoken out in the field to the IBM's S2S system during live evaluation. TestB contains sentences spoken in an office environment to the live S2S system. Using on-the-spot speakers for TestA and TestB meant to have shorter and clean sentences. Finally TestC contains pre-recorded sentences with much more hesitations and more casual conversations compared to the other two testsets. TestA, TestB and TestC have 309, 320 and 561 sentences, respectively.

LM	TestA	TestB	TestC	All
Word-3gr	18.7	18.6	38.9	32.9
TMLM	18.8	18.9	38.2	32.5
TMLM + Word-3gr	17.6	18.0	37.4	31.9

Table 1: Speech Recognition Language Model Rescoring Results.

In order to evaluate the performance of the TMLM, a lattice with a low oracle error rate was generated by a Viterbi decoder using the word trigram model (Word-3gr) model. From the lattice at most 30 (N=30) sentences are extracted for each utterance to form an N-best list. The N–best error rate for the combined test set (All) is 22.7%. The N–best size is limited (it is not in the hundreds), simply because of faster experiment turn-around. These utterances are rescored using TMLM. The results are presented in Table 1. The first two rows in the table show the baseline numbers for the word trigram (Word–3gr) model. TestA has a WER of 18.7% similar to that of TestB (18.6%). The WER for TestC is relatively high (38.9%), because, as explained above, TestC contains causal conversation with hesitations and repairs, and speakers do not necessarily stick to the domain. Moreover, when users are speaking to a device, as in the case of TestA and TestB, they use clear and shorter sentences, which are easier to recognize. The TMLM does not provide improvements for TestA and TestB but it improves the WER by 0.7% for TestC. The combined overall result is a 0.4% improvement over baseline. This improvement is not statistically significant. However, interpolating TMLM with Word-3gr improves the WER to 31.9%, which is 1.0% better than that of the Word-3gr. Standard p-test (Matched Pairs Sentence-Segment Word Error test available in standard SCLITEs statistical system comparison program from NIST) shows that this improvement is significant at $p < 0.05$ level. The interpolation weights are set equally to 0.5 for each LM.

6 Discussions

Despite limited but encouraging experimental results, we believe that the proposed perspective is a radical departure from the traditional n–gram based language modeling methods. The new perspective opens up a number of avenues which are impossible to explore in one paper.

We realize that there are a number of outstanding issues with the proposed perspective that require a closer look. We make a number of decisions to build a language model within this perspective. The decisions are sometimes ad hoc. The decisions are made in order to build a working system and are by no means the best decisions. In fact, it is quite likely that a different set of decisions may result into a better system. Using a word–morpheme co–occurrence matrix instead of a morpheme–morpheme co–occurrence matrix is one such decision. Another one is the clustering/tying of the rarely observed events to achieve robust parameter estimation both for the SVD and TMLM parameter estimation. We also use a trivial decision tree to build the models where there were no context questions. Clustering morphemes with respect to their syntactic and semantic context is another area which should be explored. In fact, we are in the process of building these models. Once we have realized the full potential of the baseline maximum likelihood TMLM, then we will investigate the discriminative training methods such as MPE (Povey & Woodland, 2002) to further improve the language model performance and adaptation to new domains using MLLR (Legetter & Woodland, 1995).

We also realize that different problems such as segmentation (e.g. Chinese) of words or morphological decomposition of words into morphemes can be addressed within the proposed perspective.

7 Conclusions

We presented our progress in improving continuous-space language modeling. We proposed the Tied-Mixture Language Model (TMLM), which allows for robust parameter estimation through the use of tying and improves on the previously presented GMLM. The new formulation lets us train a parametric language model using off–the–shelf acoustic model training tools. Our initial experimental results validated the proposed approach with encouraging results.

References

M.Afify, O. Siohan and R. Sarikaya. 2007. *Gaussian Mixture Language Models for Speech Recognition*, ICASSP, Honolulu, Hawaii.

C.J. Legetter and P.C. Woodland. 1995. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models, *Computer Speech and Language*, vol.9, pp. 171-185.

J. Bellegarda. 2000. Large Vocabulary Speech Recognition with Multispan Language Models, *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 76-84.

H. Schwenk, and J.L. Gauvain. 2003. Using Continuous Space Language Models for Conversational Telephony Speech Recognition, *IEEE Workshop on Spontaneous Speech Processing and Recognition*, Tokyo, Japan.

J. Duchateau, K. Demuynck, D.V. Compernolle and P. Wambacq. 1998. Improved Parameter Tying for Efficient Acoustic Model Evaluation in Large Vocabulary Continuous Speech Recognition. *Proc. of ICSLP*, Sydney, Australia.

Y. Gao, L. Gu, B. Zhou, R. Sarikaya, H.-K. Kuo. A.-V.I. Rosti, M. Afify, W. Zhu. 2006. IBM MASTOR: Multilingual Automatic Speech-to-Speech Translator. *Proc. of ICASSP*, Toulouse, France.

S. Chen, J. Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling, *ACL*, Santa Cruz, CA.

J. Bellagarda and D. Nahamoo. 1989. Tied mixture continuous parameter models for large vocabulary isolated speech recognition, *Proc. of ICASSP*, pp. 13-16.

X.D. Huang and M.A. Jack. 1988. Hidden Markov Modelling of Speech Based on a Semicontinuous Model, *Electronic Letters*, 24(1), pp. 6-7, 1988.

D. Povey and P.C. Woodland. 2002. Minimum phone error and I-smoothing for improved discriminative training, *Proc. of ICASSP*, pp. 105–108, Orlando, Florida.

D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, G. Zweig. 2005. fMPE: Discriminatively Trained Features for Speech Recognition, *Proc. of ICASSP*, pp. 961–964, Philadelphia, PA.

C. Chelba and F. Jelinek. 2000. Structured language modeling, *Computer Speech and Language*, 14(4), 283–332, 2000.

H. Erdogan, R. Sarikaya, S.F. Chen, Y. Gao and M. Picheny. 2005. Using Semantic Analysis to Improve Speech Recognition Performance, *Computer Speech & Language Journal*, vol. 19(3), pp: 321–343.

B. Roark, M. Saraclar, M. Collins. 2007. Using Semantic Analysis to Improve Speech Recognition Performance, *Computer Speech & Language*, vol. 21(2), pp: 373–392.

M. Riley, E. Bocchieri, A. Ljolje and M. Saraclar. 2007. The AT&T 1x real-time Switchboard speech-to-text system, *NIST RT02 Workshop*, Vienna, Virginia.

Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, vol. 3, 11371155.

S. Deerwester, Susan Dumais, G. W. Furnas, T. K. Landauer, R. Harshman. 1990. Indexing by Latent Semantic Analysis, *Journal of the American Society for Information Science*, 41 (6): 391–407.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. 1970. A Maximization Techniques Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains, *The Annals of Mathematical Statistics*, 41(1):164–171.

Shrinking Exponential Language Models

Stanley F. Chen
IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598
stanchen@watson.ibm.com

Abstract

In (Chen, 2009), we show that for a variety of language models belonging to the exponential family, the test set cross-entropy of a model can be accurately predicted from its training set cross-entropy and its parameter values. In this work, we show how this relationship can be used to motivate two heuristics for "shrinking" the size of a language model to improve its performance. We use the first heuristic to develop a novel class-based language model that outperforms a baseline word trigram model by 28% in perplexity and 1.9% absolute in speech recognition word-error rate on Wall Street Journal data. We use the second heuristic to motivate a regularized version of minimum discrimination information models and show that this method outperforms other techniques for domain adaptation.

1 Introduction

An exponential model $p_\Lambda(y|x)$ is a model with a set of features $\{f_1(x,y),\ldots,f_F(x,y)\}$ and equal number of parameters $\Lambda = \{\lambda_1,\ldots,\lambda_F\}$ where

$$p_\Lambda(y|x) = \frac{\exp(\sum_{i=1}^F \lambda_i f_i(x,y))}{Z_\Lambda(x)} \quad (1)$$

and where $Z_\Lambda(x)$ is a normalization factor. In (Chen, 2009), we show that for many types of exponential language models, if a training and test set are drawn from the same distribution, we have

$$\mathcal{H}_{\text{test}} \approx \mathcal{H}_{\text{train}} + \frac{\gamma}{D}\sum_{i=1}^F |\tilde{\lambda}_i| \quad (2)$$

where $\mathcal{H}_{\text{test}}$ denotes test set cross-entropy; $\mathcal{H}_{\text{train}}$ denotes training set cross-entropy; D is the number of events in the training data; the $\tilde{\lambda}_i$ are *regularized* parameter estimates; and γ is a constant independent of domain, training set size, and model type.[1] This relationship is strongest if the $\tilde{\Lambda} = \{\tilde{\lambda}_i\}$ are estimated using $\ell_1+\ell_2^2$ regularization (Kazama and Tsujii, 2003). In $\ell_1 + \ell_2^2$ regularization, parameters are chosen to optimize

$$\mathcal{O}_{\ell_1+\ell_2^2}(\Lambda) = \mathcal{H}_{\text{train}} + \frac{\alpha}{D}\sum_{i=1}^F |\lambda_i| + \frac{1}{2\sigma^2 D}\sum_{i=1}^F \lambda_i^2 \quad (3)$$

for some α and σ. With $(\alpha = 0.5, \sigma^2 = 6)$ and taking $\gamma = 0.938$, test set cross-entropy can be predicted with eq. (2) for a wide range of models with a mean error of a few hundredths of a nat, equivalent to a few percent in perplexity.[2]

In this paper, we show how eq. (2) can be applied to improve language model performance. First, we use eq. (2) to analyze backoff features in exponential n-gram models. We find that backoff features improve test set performance by reducing the "size" of a model $\frac{1}{D}\sum_{i=1}^F |\tilde{\lambda}_i|$ rather than by improving training set performance. This suggests the following principle for improving exponential language models: if a model can be "shrunk" without increasing its training set cross-entropy, test set cross-entropy should improve. We apply this idea to motivate two language models: a novel class-based language model and regularized minimum discrimination information (MDI) models. We show how these models outperform other models in both perplexity and word-error rate on Wall Street Journal (WSJ) data.

The organization of this paper is as follows: In Section 2, we analyze the use of backoff features in n-gram models to motivate a heuristic for model design. In Sections 3 and 4, we introduce our novel

[1] The cross-entropy of a model $p_\Lambda(y|x)$ on some data $\mathcal{D} = (x_1,y_1),\ldots,(x_D,y_D)$ is defined as $-\frac{1}{D}\sum_{j=1}^D \log p_\Lambda(y_j|x_j)$. It is equivalent to the negative mean log-likelihood per event as well as to log perplexity.

[2] A *nat* is a "natural" bit and is equivalent to $\log_2 e$ regular bits. We use nats to be consistent with (Chen, 2009).

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 468–476,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

features	$\mathcal{H}_{\text{eval}}$	$\mathcal{H}_{\text{pred}}$	$\mathcal{H}_{\text{train}}$	$\sum \frac{\lvert\tilde{\lambda}_i\rvert}{D}$
3g	2.681	2.724	2.341	0.408
2g+3g	2.528	2.513	2.248	0.282
1g+2g+3g	2.514	2.474	2.241	0.249

Table 1: Various statistics for letter trigram models built on a 1k-word training set. $\mathcal{H}_{\text{eval}}$ is the cross-entropy of the evaluation data; $\mathcal{H}_{\text{pred}}$ is the predicted test set cross-entropy according to eq. (2); and $\mathcal{H}_{\text{train}}$ is the training set cross-entropy. The evaluation data is drawn from the same distribution as the training; $\mathcal{H}$ values are in nats.

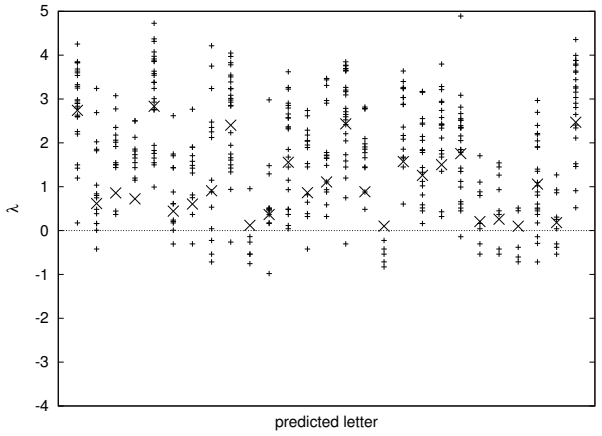


Figure 1: Nonzero $\tilde{\lambda}_i$ values for bigram features in letter bigram model without unigram backoff features. If we denote bigrams as $w_{j-1}w_j$, each column contains the $\tilde{\lambda}_i$'s corresponding to all bigrams with a particular w_j. The '$\times$' marks represent the average $|\tilde{\lambda}_i|$ in each column; this average includes history words for which no feature exists or for which $\tilde{\lambda}_i = 0$.

class-based model and discuss MDI domain adaptation, and compare these methods against other techniques on WSJ data. Finally, in Sections 5 and 6 we discuss related work and conclusions.[3]

2 N-Gram Models and Backoff Features

In this section, we use eq. (2) to explain why backoff features in exponential n-gram models improve performance, and use this analysis to motivate a general heuristic for model design. An exponential n-gram model contains a binary feature f_ω for each n'-gram ω occurring in the training data for $n' \leq n$, where $f_\omega(x, y) = 1$ iff xy ends in ω. We refer to features corresponding to n'-grams for $n' < n$ as *backoff* features; it is well known that backoff features help

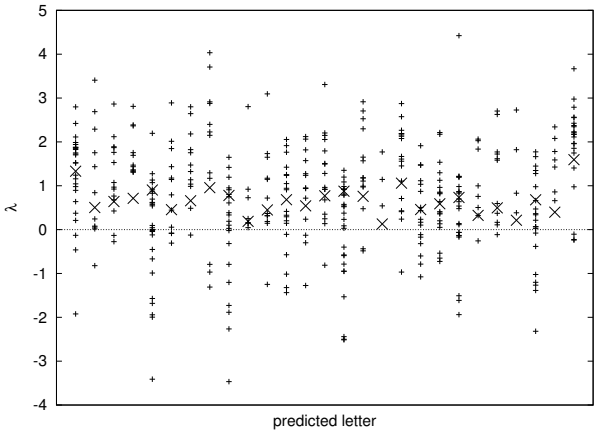


Figure 2: Like Figure 1, but for model with unigram backoff features.

performance a great deal. We present statistics in Table 1 for various letter trigram models built on the same data set. In these and all later experiments, all models are regularized with $\ell_1 + \ell_2^2$ regularization with $(\alpha = 0.5, \sigma^2 = 6)$. The last row corresponds to a normal trigram model; the second row corresponds to a model lacking unigram features; and the first row corresponds to a model with no unigram or bigram features. As backoff features are added, we see that the training set cross-entropy improves, which is not surprising since the number of features is increasing. More surprising is that as we add features, the "size" of the model $\frac{1}{D}\sum_{i=1}^{F} |\tilde{\lambda}_i|$ decreases.

We can explain these results by examining a simple example. Consider an exponential model consisting of the features $f_1(x, y)$ and $f_2(x, y)$ with parameter values $\tilde{\lambda}_1 = 3$ and $\tilde{\lambda}_2 = 4$. From eq. (1), this model has the form

$$p_{\tilde{\Lambda}}(y|x) = \frac{\exp(3f_1(x, y) + 4f_2(x, y))}{Z_\Lambda(x)} \qquad (4)$$

Now, consider creating a new feature $f_3(x, y) = f_1(x, y) + f_2(x, y)$ and setting our parameters as follows: $\lambda_1^{\text{new}} = 0$, $\lambda_2^{\text{new}} = 1$, and $\lambda_3^{\text{new}} = 3$. Substituting into eq. (1), we see that $p_{\Lambda^{\text{new}}}(y|x) = p_{\tilde{\Lambda}}(y|x)$ for all x, y. As the distribution this model describes does not change, neither will its training performance. However, the (unscaled) size $\sum_{i=1}^{F} |\lambda_i|$ of the model has been reduced from 3+4=7 to 0+1+3=4, and consequently by eq. (2) we predict that test performance will improve.[4]

[3] A long version of this paper can be found at (Chen, 2008).

[4] When $\text{sgn}(\tilde{\lambda}_1) = \text{sgn}(\tilde{\lambda}_2)$, $\sum_{i=1}^{F} |\lambda_i|$ is reduced most by

In fact, since $p_{\Lambda^{\text{new}}} = p_{\tilde{\Lambda}}$, test performance will remain the same. The catch is that eq. (2) applies only to the *regularized* parameter estimates for a model, and in general, Λ^{new} will not be the regularized parameter estimates for the expanded feature set. We can compute the actual regularized parameters $\tilde{\Lambda}^{\text{new}}$ for which eq. (2) *will* apply; this may improve predicted performance even more.

Hence, by adding "redundant" features to a model to shrink its total size $\sum_{i=1}^{F} |\tilde{\lambda}_i|$, we can improve predicted performance (and perhaps also actual performance). This analysis suggests the following technique for improving model performance:

Heuristic 1 *Identify groups of features which will tend to have similar $\tilde{\lambda}_i$ values. For each such feature group, add a new feature to the model that is the sum of the original features.*

The larger the original $\tilde{\lambda}_i$'s, the larger the reduction in model size and the higher the predicted gain.

Given this perspective, we can explain why backoff features improve n-gram model performance. For simplicity, consider a bigram model, one without unigram backoff features. It seems intuitive that probabilities of the form $p(w_j|w_{j-1})$ are similar across different w_{j-1}, and thus so are the $\tilde{\lambda}_i$ for the corresponding bigram features. (If a word has a high unigram probability, it will also tend to have high bigram probabilities.) In Figure 1, we plot the nonzero $\tilde{\lambda}_i$ values for all (bigram) features in a bigram model without unigram features. Each column contains the $\tilde{\lambda}_i$ values for a different predicted word w_j, and the '$\times$' mark in each column is the average value of $|\tilde{\lambda}_i|$ over all history words w_{j-1}. We see that the average $|\tilde{\lambda}_i|$ for each word w_j is often quite far from zero, which suggests creating features

$$f_{w_j}(x, y) = \sum_{w_{j-1}} f_{w_{j-1}w_j}(x, y) \tag{5}$$

to reduce the overall size of the model.

In fact, these features are exactly unigram backoff features. In Figure 2, we plot the nonzero $\tilde{\lambda}_i$ values for all bigram features after adding unigram backoff features. We see that the average $|\tilde{\lambda}_i|$'s are closer to zero, implying that the model size $\sum_{i=1}^{F} |\tilde{\lambda}_i|$ has been significantly decreased. We can extend this idea to higher-order n-gram models as well; *e.g.*, bigram parameters can shrink trigram parameters, and can in turn be shrunk by unigram parameters. As shown in Table 1, both training set cross-entropy and model size can be reduced by this technique.

setting λ_3^{new} to the $\tilde{\lambda}_i$ with the smaller magnitude, and the size of the reduction is equal to $|\lambda_3^{\text{new}}|$. If $\text{sgn}(\tilde{\lambda}_1) \neq \text{sgn}(\tilde{\lambda}_2)$, no reduction is possible through this transformation.

	$\mathcal{H}_{\text{eval}}$	$\mathcal{H}_{\text{pred}}$	$\mathcal{H}_{\text{train}}$	$\sum \frac{\lvert\tilde{\lambda}_i\rvert}{D}$
word n-gram	4.649	4.672	3.354	1.405
model **M**	4.536	4.544	3.296	1.330

Table 2: Various statistics for word and class trigram models built on 100k sentences of WSJ training data.

3 Class-Based Language Models

In this section, we show how we can use Heuristic 1 to design a novel class-based model that outperforms existing models in both perplexity and speech recognition word-error rate. We assume a word w is always mapped to the same class $c(w)$. For a sentence $w_1 \cdots w_l$, we have

$$p(w_1 \cdots w_l) = \prod_{j=1}^{l+1} p(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) \times \prod_{j=1}^{l} p(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) \tag{6}$$

where $c_j = c(w_j)$ and c_{l+1} is the end-of-sentence token. We use the notation $p_{\text{ng}}(y|\omega)$ to denote an exponential n-gram model, a model containing a feature for each suffix of each ωy occurring in the training set. We use $p_{\text{ng}}(y|\omega_1, \omega_2)$ to denote a model containing all features in $p_{\text{ng}}(y|\omega_1)$ and $p_{\text{ng}}(y|\omega_2)$.

We can define a class-based n-gram model by choosing parameterizations for the distributions $p(c_j|\cdots)$ and $p(w_j|\cdots)$ in eq. (6) above. For example, the most widely-used class-based n-gram model is the one introduced by Brown et al. (1992); we refer to this model as the IBM class model:

$$\begin{aligned} p(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) &= p_{\text{ng}}(c_j|c_{j-2}c_{j-1}) \\ p(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) &= p_{\text{ng}}(w_j|c_j) \end{aligned} \tag{7}$$

(In the original work, non-exponential n-gram models are used.) Clearly, there is a large space of possible class-based models.

Now, we discuss how we can use Heuristic 1 to design a novel class-based model by using class information to "shrink" a word-based n-gram model. The basic idea is as follows: if we have an n-gram ω

and another n-gram ω' created by replacing a word in ω with a similar word, then the two corresponding features should have similar $\tilde{\lambda}_i$'s. For example, it seems intuitive that the n-grams *on Monday morning* and *on Tuesday morning* should have similar $\tilde{\lambda}_i$'s. Heuristic 1 tells us how to take advantage of this observation to improve model performance.

Let's begin with a word trigram model $p_{\text{ng}}(w_j|w_{j-2}w_{j-1})$. First, we would like to convert this model into a class-based model. Without loss of generality, we have

$$\begin{aligned} p(w_j|w_{j-2}w_{j-1}) &= \textstyle\sum_{c_j} p(w_j, c_j|w_{j-2}w_{j-1}) \\ &= \textstyle\sum_{c_j} p(c_j|w_{j-2}w_{j-1})p(w_j|w_{j-2}w_{j-1}c_j) \end{aligned} \quad (8)$$

Thus, it seems reasonable to use the distributions $p_{\text{ng}}(c_j|w_{j-2}w_{j-1})$ and $p_{\text{ng}}(w_j|w_{j-2}w_{j-1}c_j)$ as the starting point for our class model. This model can express the same set of word distributions as our original model, and hence may have a similar training cross-entropy. In addition, this transformation can be viewed as shrinking together word n-grams that differ only in w_j. That is, we expect that pairs of n-grams $w_{j-2}w_{j-1}w_j$ that differ only in w_j (belonging to the same class) should have similar $\tilde{\lambda}_i$. From Heuristic 1, we can make new features

$$f_{w_{j-2}w_{j-1}c_j}(x, y) = \sum_{w_j \in c_j} f_{w_{j-2}w_{j-1}w_j}(x, y) \quad (9)$$

These are exactly the features in $p_{\text{ng}}(c_j|w_{j-2}w_{j-1})$. When applying Heuristic 1, all features typically belong to the same model, but even when they don't one can achieve the same net effect.

Then, we can use Heuristic 1 to also shrink together n-gram features for n-grams that differ only in their histories. For example, we can create new features of the form

$$f_{c_{j-2}c_{j-1}c_j}(x, y) = \sum_{w_{j-2} \in c_{j-2}, w_{j-1} \in c_{j-1}} f_{w_{j-2}w_{j-1}c_j}(x, y) \quad (10)$$

This corresponds to replacing $p_{\text{ng}}(c_j|w_{j-2}w_{j-1})$ with the distribution $p_{\text{ng}}(c_j|c_{j-2}c_{j-1}, w_{j-2}w_{j-1})$. We refer to the resulting model as model **M**:

$$\begin{aligned} p(c_j|c_1 \cdots c_{j-1}, w_1 \cdots w_{j-1}) &= p_{\text{ng}}(c_j|c_{j-2}c_{j-1}, w_{j-2}w_{j-1}) \\ p(w_j|c_1 \cdots c_j, w_1 \cdots w_{j-1}) &= p_{\text{ng}}(w_j|w_{j-2}w_{j-1}c_j) \end{aligned} \quad (11)$$

By design, it is meant to have similar training set cross-entropy as a word n-gram model while being significantly smaller.

To give an idea of whether this model behaves as expected, in Table 2 we provide statistics for this model (as well as for an exponential word n-gram model) built on 100k WSJ training sentences with 50 classes using the same regularization as before. We see that model **M** is both smaller than the baseline and has a lower training set cross-entropy, similar to the behavior found when adding backoff features to word n-gram models in Section 2. As long as eq. (2) holds, model **M** should have good test performance; in (Chen, 2009), we show that eq. (2) does indeed hold for models of this type.

3.1 Class-Based Model Comparison

In this section, we compare model **M** against other class-based models in perplexity and word-error rate. The training data is 1993 WSJ text with verbalized punctuation from the CSR-III Text corpus, and the vocabulary is the union of the training vocabulary and 20k-word "closed" test vocabulary from the first WSJ CSR corpus (Paul and Baker, 1992). We evaluate training set sizes of 1k, 10k, 100k, and 900k sentences. We create three different word classings containing 50, 150, and 500 classes using the algorithm of Brown et al. (1992) on the largest training set.[5] For each training set and number of classes, we build 3-gram and 4-gram versions of each model.

From the verbalized punctuation data from the training and test portions of the WSJ CSR corpus, we randomly select 2439 unique utterances (46888 words) as our evaluation set. From the remaining verbalized punctuation data, we select 977 utterances (18279 words) as our development set.

We compare the following model types: conventional (*i.e.*, non-exponential) word n-gram models; conventional IBM class n-gram models interpolated with conventional word n-gram models (Brown et al., 1992); and model **M**. All conventional n-gram models are smoothed with modified Kneser-Ney smoothing (Chen and Goodman, 1998), except we also evaluate word n-gram models with Katz smoothing (Katz, 1987). *Note:* Because word

[5] One can imagine choosing word classes to optimize model shrinkage; however, this is not an avenue we pursued.

	training set (sents.)			
	1k	10k	100k	900k
conventional word n-gram, Katz				
3g	**579.3**	**317.1**	**196.7**	137.5
4g	592.6	325.6	202.4	**136.7**
interpolated IBM class model				
3g, 50c	358.4	224.5	156.8	117.8
3g, 150c	346.5	210.5	149.0	114.7
3g, 500c	372.6	210.9	145.8	112.3
4g, 50c	362.1	220.4	149.6	109.1
4g, 150c	**346.3**	**207.8**	142.5	105.2
4g, 500c	371.5	207.9	**140.5**	**103.6**

	training set (sents.)			
	1k	10k	100k	900k
conventional word n-gram, modified KN				
3g	488.4	270.6	168.2	121.5
4g	**486.8**	**267.4**	**163.6**	**114.4**
model **M**				
3g, 50c	**341.5**	210.0	144.5	110.9
3g, 150c	342.6	203.7	140.0	108.0
3g, 500c	387.5	212.7	142.2	108.1
4g, 50c	345.8	209.0	139.1	101.6
4g, 150c	344.1	**202.8**	**135.7**	**99.1**
4g, 500c	390.7	211.1	138.5	100.6

Table 3: WSJ perplexity results. The best performance for each training set for each model type is highlighted in bold.

	training set (sents.)			
	1k	10k	100k	900k
conventional word n-gram, Katz				
3g	**35.5%**	**30.7%**	**26.2%**	**22.7%**
4g	35.6%	30.9%	26.3%	**22.7%**
interpolated IBM class model				
3g, 50c	32.2%	28.7%	25.2%	22.5%
3g, 150c	**31.8%**	28.1%	25.0%	22.3%
3g, 500c	32.5%	28.5%	**24.5%**	22.1%
4g, 50c	32.2%	28.6%	25.0%	22.0%
4g, 150c	**31.8%**	**28.0%**	24.6%	21.8%
4g, 500c	32.7%	28.3%	**24.5%**	**21.6%**

	training set (sents.)			
	1k	10k	100k	900k
conventional word n-gram, modified KN				
3g	**34.5%**	30.5%	26.1%	22.6%
4g	**34.5%**	**30.4%**	**25.7%**	**22.3%**
model **M**				
3g, 50c	**30.8%**	27.4%	24.0%	21.7%
3g, 150c	31.0%	**27.1%**	23.8%	21.5%
3g, 500c	32.3%	27.8%	23.9%	21.4%
4g, 50c	**30.8%**	27.5%	23.9%	21.2%
4g, 150c	31.0%	**27.1%**	**23.5%**	**20.8%**
4g, 500c	32.4%	27.9%	24.1%	21.1%

Table 4: WSJ lattice rescoring results; all values are word-error rates. The best performance for each training set size for each model type is highlighted in bold. Each 0.1% in error rate corresponds to about 47 errors.

classes are derived from the largest training set, results for word models and class models are comparable only for this data set. The interpolated model is the most popular state-of-the-art class-based model in the literature, and is the only model here using the development set to tune interpolation weights.

We display the perplexities of these models on the evaluation set in Table 3. Model **M** performs best of all (even without interpolating with a word n-gram model), outperforming the interpolated model with every training set and achieving its largest reduction in perplexity (4%) on the largest training set. While these perplexity reductions are quite modest, what matters more is speech recognition performance.

For the speech recognition experiments, we use a cross-word quinphone system built from 50 hours of Broadcast News data. The system contains 2176 context-dependent states and a total of 50336 Gaussians. To evaluate our language models, we use lattice rescoring. We generate lattices on both our development and evaluation data sets using the Latt-AIX decoder (Saon et al., 2005) in the Attila speech recognition system (Soltau et al., 2005). The language model for lattice generation is created by building a modified Kneser-Ney-smoothed word trigram model on our largest training set; this model is pruned to contain a total of 350k n-grams using the algorithm of Stolcke (1998). We choose the acoustic weight for each model to optimize word-error rate on the development set.

In Table 4, we display the word-error rates for each model. If we compare the best performance of model **M** for each training set with that of the state-of-the-art interpolated class model, we find that model **M** is superior by 0.8–1.0% absolute. These gains are much larger than are suggested by the perplexity gains of model **M** over the interpolated model; as has been observed earlier, perplexity is

	$\mathcal{H}_{\text{eval}}$	$\mathcal{H}_{\text{pred}}$	$\mathcal{H}_{\text{train}}$	$\sum \frac{\lvert\tilde{\lambda}_i\rvert}{D}$
baseline n-gram model				
1k	5.915	5.875	2.808	3.269
10k	5.212	5.231	3.106	2.265
100k	4.649	4.672	3.354	1.405
MDI n-gram model				
1k	5.444	5.285	2.678	2.780
10k	5.031	4.973	3.053	2.046
100k	4.611	4.595	3.339	1.339

Table 5: Various statistics for WSJ trigram models, with and without a Broadcast News prior model. The first column is the size of the in-domain training set in sentences.

not a reliable predictor of speech recognition performance. While we can only compare class models with word models on the largest training set, for this training set model **M** outperforms the baseline Katz-smoothed word trigram model by 1.9% absolute.[6]

4 Domain Adaptation

In this section, we introduce another heuristic for improving exponential models and show how this heuristic can be used to motivate a regularized version of minimum discrimination information (MDI) models (Della Pietra et al., 1992). Let's say we have a model $p_{\tilde{\Lambda}}$ estimated from one training set and a "similar" model q estimated from an independent training set. Imagine we use q as a *prior* model for p_Λ; *i.e.*, we make a new model $p^q_{\Lambda^{\text{new}}}$ as follows:

$$p^q_{\Lambda^{\text{new}}}(y|x) = q(y|x)\frac{\exp(\sum_{i=1}^F \lambda_i^{\text{new}} f_i(x,y))}{Z_{\Lambda^{\text{new}}}(x)} \quad (12)$$

Then, choose Λ^{new} such that $p^q_{\Lambda^{\text{new}}}(y|x) = p_{\tilde{\Lambda}}(y|x)$ for all x, y (assuming this is possible). If q is "similar" to $p_{\tilde{\Lambda}}$, then we expect the size $\frac{1}{D}\sum_{i=1}^F |\lambda_i^{\text{new}}|$ of $p^q_{\Lambda^{\text{new}}}$ to be less than that of $p_{\tilde{\Lambda}}$. Since they describe the same distribution, their training set cross-entropy will be the same. By eq. (2), we expect $p^q_{\Lambda^{\text{new}}}$ to have better test set performance than $p_{\tilde{\Lambda}}$ after reestimation.[7] In (Chen, 2009), we show that eq. (2) does indeed hold for models with priors; q need not be accounted for in computing model size as long as it is estimated on a separate training set.

[6] Results for several other baseline language models and with a different acoustic model are given in (Chen, 2008).

[7] That is, we expect the *regularized* parameters $\tilde{\Lambda}^{\text{new}}$ to yield improved performance.

This analysis suggests the following method for improving model performance:

Heuristic 2 *Find a "similar" distribution estimated from an independent training set, and use this distribution as a prior.*

It is straightforward to apply this heuristic to the task of domain adaptation for language modeling. In the usual formulation of this task, we have a test set and a small training set from the same domain, and a large training set from a different domain. The goal is to use the data from the outside domain to maximally improve language modeling performance on the target domain. By Heuristic 2, we can build a language model on the outside domain, and use this model as the prior model for a language model built on the in-domain data. This method is identical to the MDI method for domain adaptation, except that we also apply regularization.

In our domain adaptation experiments, our out-of-domain data is a 100k-sentence Broadcast News training set. For our in-domain WSJ data, we use training set sizes of 1k, 10k, and 100k sentences. We build an exponential n-gram model on the Broadcast News data and use this model as the prior model $q(y|x)$ in eq. (12) when building an exponential n-gram model on the in-domain data. In Table 5, we display various statistics for trigram models built on varying amounts of in-domain data when using a Broadcast News prior and not. Across training sets, the MDI models are both smaller in $\frac{1}{D}\sum_{i=1}^F |\tilde{\lambda}_i|$ and have better training set cross-entropy than the unadapted models built on the same data. By eq. (2), the adapted models should have better test performance and we verify this in the next section.

4.1 Domain Adaptation Method Comparison

In this section, we examine how MDI adaptation compares to other state-of-the-art methods for domain adaptation in both perplexity and speech recognition word-error rate. For these experiments, we use the same development and evaluation sets and lattice rescoring setup from Section 3.1.

The most widely-used techniques for domain adaptation are linear interpolation and count merging. In linear interpolation, separate n-gram models are built on the in-domain and out-of-domain data and are interpolated together. In count merging, the

	in-domain data (sents.)			in-domain data (sents.)		
	1k	10k	100k	1k	10k	100k
in-domain data only						
3g	488.4	270.6	168.2	**34.5%**	30.5%	26.1%
4g	**486.8**	**267.4**	**163.6**	**34.5%**	**30.4%**	**25.7%**
count merging						
3g	503.1	290.9	170.7	30.4%	28.3%	**25.2%**
4g	**497.1**	**284.9**	**165.3**	**30.0%**	**28.0%**	25.3%
linear interpolation						
3g	328.3	234.8	162.6	**30.3%**	28.5%	25.8%
4g	**325.3**	**230.8**	**157.6**	**30.3%**	**28.4%**	**25.2%**
MDI model						
3g	296.3	218.7	157.0	30.0%	28.0%	**24.9%**
4g	**293.7**	**215.8**	**152.5**	**29.6%**	**27.9%**	**24.9%**

Table 6: WSJ perplexity and lattice rescoring results for domain adaptation models. Values on the left are perplexities and values on the right are word-error rates.

in-domain and out-of-domain data are concatenated into a single training set, and a single n-gram model is built on the combined data set. The in-domain data set may be replicated several times to more heavily weight this data. We also consider the baseline of not using the out-of-domain data.

In Table 6, we display perplexity and word-error rates for each method, for both trigram and 4-gram models and with varying amounts of in-domain training data. The last method corresponds to the exponential MDI model; all other methods employ conventional (non-exponential) n-gram models with modified Kneser-Ney smoothing. In count merging, only one copy of the in-domain data is included in the training set; including more copies does not improve evaluation set word-error rate.

Looking first at perplexity, MDI models outperform the next best method, linear interpolation, by about 10% in perplexity on the smallest data set and 3% in perplexity on the largest. In terms of word-error rate, MDI models again perform best of all, outperforming interpolation by 0.3–0.7% absolute and count merging by 0.1–0.4% absolute.

5 Related Work

5.1 Class-Based Language Models

In past work, the most common baseline models are Katz-smoothed word trigram models. Compared to this baseline, model **M** achieves a perplexity reduction of 28% and word-error rate reduction of 1.9% absolute with a 900k-sentence training set. The most closely-related existing model to model **M** is the model *fullibmpredict* proposed by Goodman (2001):

$$\begin{aligned} &p(c_j|c_{j-2}c_{j-1},w_{j-2}w_{j-1})= \\ &\qquad \lambda\, p(c_j|w_{j-2}w_{j-1})+(1-\lambda)\, p(c_j|c_{j-2}c_{j-1}) \\ &p(w_j|c_{j-2}c_{j-1}c_j,w_{j-2}w_{j-1})= \\ &\qquad \mu\, p(w_j|w_{j-2}w_{j-1}c_j)+(1-\mu)\, p(w_j|c_{j-2}c_{j-1}c_j) \end{aligned} \quad (13)$$

This is similar to model **M** except that linear interpolation is used to combine word and class history information, and there is no analog to the final term in eq. (13) in model **M**. Using the North American Business news corpus, the largest perplexity reduction achieved over a Katz-smoothed trigram model baseline by *fullibmpredict* is about 25%, with a training set of 1M words. In N-best list rescoring with a 284M-word training set, the best result achieved for an individual class-based model is an 0.5% absolute reduction in word-error rate.

To situate the quality of our results, we also review the best perplexity and word-error rate results reported for class-based language models relative to conventional word n-gram model baselines. In terms of absolute word-error rate, the best gains we found in the literature are from *multi-class composite* n-gram models, a variant of the IBM class model (Yamamoto and Sagisaka, 1999; Yamamoto et al., 2003). These are called *composite* models because frequent word sequences can be concatenated into single units within the model; the term *multi-class* refers to choosing different word clusterings depending on word position. In experiments on the ATR spoken language database, Yamamoto et al. (2003) report a reduction in perplexity of 9% and an increase in word accuracy of 2.2% absolute over a Katz-smoothed trigram model.

In terms of perplexity, the best gains we found are from SuperARV language models (Wang and Harper, 2002; Wang et al., 2002; Wang et al., 2004). In these models, classes are based on *abstract role values* as given by a Constraint Dependency Grammar. The class and word prediction distributions are n-gram models that back off to a variety of mixed word/class histories in a specific order. With a WSJ training set of 37M words and a Katz-smoothed trigram model baseline, a perplexity reduction of up to

53% is achieved as well as a decrease in word-error rate of up to 1.0% absolute.

All other perplexity and absolute word-error rate gains we found in the literature are considerably smaller than those listed here. While different data sets are used in previous work so results are not directly comparable, our results appear very competitive with the body of existing results in the literature.

5.2 Domain Adaptation

Here, we discuss methods for supervised domain adaptation that involve only the simple static combination of in-domain and out-of-domain data or models. For a survey of techniques using word classes, topic, syntax, etc., refer to (Bellegarda, 2004).

Linear interpolation is the most widely-used method for domain adaptation. Jelinek et al. (1991) describe its use for combining a cache language model and static language model. Another popular method is count merging; this has been motivated as an instance of MAP adaptation (Federico, 1996; Masataki et al., 1997). In terms of word-error rate, Iyer et al. (1997) found linear interpolation to give better speech recognition performance while Bacchiani et al. (2006) found count merging to be superior. Klakow (1998) proposes log-linear interpolation for domain adaptation. As compared to regular linear interpolation for bigram models, an improvement of 4% in perplexity and 0.2% absolute in word-error rate is found.

Della Pietra et al. (1992) introduce the idea of minimum discrimination information distributions. Given a prior model $q(y|x)$, the goal is to find the nearest model in Kullback-Liebler divergence that satisfies a set of linear constraints derived from adaptation data. The model satisfying these conditions is an exponential model containing one feature per constraint with $q(y|x)$ as its prior as in eq. (12). While MDI models have been used many times for language model adaptation, *e.g.*, (Kneser et al., 1997; Federico, 1999), they have not performed as well as linear interpolation in perplexity or word-error rate (Rao et al., 1995; Rao et al., 1997).

One important issue with MDI models is how to select the feature set specifying the model. With a small amount of adaptation data, one should intuitively use a small feature set, *e.g.*, containing just unigram features. However, the use of regularization can obviate the need for intelligent feature selection. In this work, we include all n-gram features present in the adaptation data for $n \in \{3, 4\}$. Chueh and Chien (2008) propose the use of inequality constraints for regularization (Kazama and Tsujii, 2003); here, we use $\ell_1 + \ell_2^2$ regularization instead. We hypothesize that the use of state-of-the-art regularization is the primary reason why we achieve better performance relative to interpolation and count merging as compared to earlier work.

6 Discussion

For exponential language models, eq. (2) tells us that with respect to test set performance, the number of model parameters seems to matter not at all; all that matters are the magnitudes of the parameter values. Consequently, one can improve exponential language models by adding features (or a prior model) that shrink parameter values while maintaining training performance, and from this observation we develop Heuristics 1 and 2. We use these ideas to motivate a novel and simple class-based language model that achieves perplexity and word-error rate improvements competitive with the best reported results for class-based models in the literature. In addition, we show that with regularization, MDI models can outperform both linear interpolation and count merging in language model combination. Still, Heuristics 1 and 2 are quite vague, and it remains to be seen how to determine when these heuristics will be effective.

In summary, we have demonstrated how the trade-off between training set performance and model size impacts aspects of language modeling as diverse as backoff n-gram features, class-based models, and domain adaptation. In particular, we can frame performance improvements in all of these areas as methods that shrink models without degrading training set performance. All in all, eq. (2) is an important tool for both understanding and improving language model performance.

Acknowledgements

We thank Bhuvana Ramabhadran and the anonymous reviewers for their comments on this and earlier versions of the paper.

References

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

Jerome R. Bellegarda. 2004. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42(1):93–108.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University.

Stanley F. Chen. 2008. Performance prediction for exponential language models. Technical Report RC 24671, IBM Research Division, October.

Stanley F. Chen. 2009. Performance prediction for exponential language models. In *Proc. of HLT-NAACL*.

Chuang-Hua Chueh and Jen-Tzung Chien. 2008. Reliable feature selection for language model adaptation. In *Proc. of ICASSP*, pp. 5089–5092.

Stephen Della Pietra, Vincent Della Pietra, Robert L. Mercer, and Salim Roukos. 1992. Adaptive language modeling using minimum discriminant estimation. In *Proc. of the Speech and Natural Language DARPA Workshop*, February.

Marcello Federico. 1996. Bayesian estimation methods for n-gram language model adaptation. In *Proc. of ICSLP*, pp. 240–243.

Marcello Federico. 1999. Efficient language model adaptation through MDI estimation. In *Proc. of Eurospeech*, pp. 1583–1586.

Joshua T. Goodman. 2001. A bit of progress in language modeling. Technical Report MSR-TR-2001-72, Microsoft Research.

Rukmini Iyer, Mari Ostendorf, and Herbert Gish. 1997. Using out-of-domain data to improve in-domain language models. *IEEE Signal Processing Letters*, 4(8):221–223, August.

Frederick Jelinek, Bernard Merialdo, Salim Roukos, and Martin Strauss. 1991. A dynamic language model for speech recognition. In *Proc. of the DARPA Workshop on Speech and Natural Language*, pp. 293–295, Morristown, NJ, USA.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, March.

Jun'ichi Kazama and Jun'ichi Tsujii. 2003. Evaluation and extension of maximum entropy models with inequality constraints. In *Proc. of EMNLP*, pp. 137–144.

Dietrich Klakow. 1998. Log-linear interpolation of language models. In *Proc. of ICSLP*.

Reinhard Kneser, Jochen Peters, and Dietrich Klakow. 1997. Language model adaptation using dynamic marginals. In *Proc. of Eurospeech*.

Hirokazu Masataki, Yoshinori Sagisaka, Kazuya Hisaki, and Tatsuya Kawahara. 1997. Task adaptation using MAP estimation in n-gram language modeling. In *Proc. of ICASSP*, volume 2, pp. 783–786, Washington, DC, USA. IEEE Computer Society.

Douglas B. Paul and Janet M. Baker. 1992. The design for the Wall Street Journal-based CSR corpus. In *Proc. of the DARPA Speech and Natural Language Workshop*, pp. 357–362, February.

P. Srinivasa Rao, Michael D. Monkowski, and Salim Roukos. 1995. Language model adaptation via minimum discrimination information. In *Proc. of ICASSP*, volume 1, pp. 161–164.

P. Srinivasa Rao, Satya Dharanipragada, and Salim Roukos. 1997. MDI adaptation of language models across corpora. In *Proc. of Eurospeech*, pp. 1979–1982.

George Saon, Daniel Povey, and Geoffrey Zweig. 2005. Anatomy of an extremely fast LVCSR decoder. In *Proc. of Interspeech*, pp. 549–552.

Hagen Soltau, Brian Kingsbury, Lidia Mangu, Daniel Povey, George Saon, and Geoffrey Zweig. 2005. The IBM 2004 conversational telephony system for rich transcription. In *Proc. of ICASSP*, pp. 205–208.

Andreas Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proc. of the DARPA Broadcast News Transcription and Understanding Workshop*, pp. 270–274, Lansdowne, VA, February.

Wen Wang and Mary P. Harper. 2002. The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proc. of EMNLP*, pp. 238–247.

Wen Wang, Yang Liu, and Mary P. Harper. 2002. Rescoring effectiveness of language models using different levels of knowledge and their integration. In *Proc. of ICASSP*, pp. 785–788.

Wen Wang, Andreas Stolcke, and Mary P. Harper. 2004. The use of a linguistically motivated language model in conversational speech recognition. In *Proc. of ICASSP*, pp. 261–264.

Hirofumi Yamamoto and Yoshinori Sagisaka. 1999. Multi-class composite n-gram based on connection direction. In *Proc. of ICASSP*, pp. 533–536.

Hirofumi Yamamoto, Shuntaro Isogai, and Yoshinori Sagisaka. 2003. Multi-class composite n-gram language model. *Speech Communication*, 41(2-3):369–379.

Predicting Response to Political Blog Posts with Topic Models

Tae Yano William W. Cohen Noah A. Smith
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{taey,wcohen,nasmith}@cs.cmu.edu

Abstract

In this paper we model discussions in online political blogs. To do this, we extend Latent Dirichlet Allocation (Blei et al., 2003), in various ways to capture different characteristics of the data. Our models jointly describe the generation of the primary documents (posts) as well as the authorship and, optionally, the contents of the blog community's verbal reactions to each post (comments). We evaluate our model on a novel comment prediction task where the models are used to predict which blog users will leave comments on a given post. We also provide a qualitative discussion about what the models discover.

1 Introduction

Web logging (blogging) and its social impact have recently attracted considerable public and scientific interest. One use of blogs is as a community discussion forum, especially for political discussion and debate. Blogging has arguably opened a new channel for huge numbers of people to express their views with unprecedented speed and to unprecedented audiences. Their collective behavior in the blogosphere has already been noted in the American political arena (Adamic and Glance, 2005). In this paper we attempt to deliver a framework useful for analyzing text in blogs quantitatively as well as qualitatively. Better blog text analysis could lead to better automated recommendation, organization, extraction, and retrieval systems, and might facilitate data-driven research in the social sciences.

Apart from the potential social utility of text processing for this domain, we believe blog data is worthy of scientific study in its own right. The spontaneous, reactive, and informal nature of the language in this domain seems to defy conventional analytical approaches in NLP such as supervised text classification (Mullen and Malouf, 2006), yet the data are rich in argumentative, topical, and temporal structure that can perhaps be modeled computationally. We are especially interested in the semi-causal structure of blog discussions, in which a post "spawns" comments (or fails to do so), which meander among topics and asides and show the personality of the participants and the community.

Our approach is to develop probabilistic models for the generation of blog posts and comments jointly within a blog site. The model is an extension of Latent Dirichlet Allocation (Blei et al., 2003). Unsupervised topic models can be applied to collections of unannotated documents, requiring very little corpus engineering. They can be easily adapted to new problems by altering the graphical model, then applying standard probabilistic inference algorithms. Different models can be compared to explore the ramifications of different hypotheses about the data. For example, we will explore whether the contents of posts a user has commented on in the past and the words she has used can help predict which posts she will respond to in the future.

The paper is organized as follows. In §2 we review prior work on topic modeling for document collections and studies of social media like political blogs. We then provide a qualitative characterization of political blogs, highlighting some of the features we believe a computational model should capture and discuss our new corpus of political blogs (§3). We present several different candidate topic models that aim to capture these ideas in §4. §5 shows our empirical evaluation on a new comment prediction task and a qualitative analysis of the models learned.

2 Related Work

Network analysis, including citation analysis, has been applied to document collections on the Web (Cohn and Hofmann, 2001). Adamic and Glance (2005) applied network analysis to the political bl-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 477–485,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

ogosphere. The study modeled the large, complex structure of the political blogosphere as a network of hyperlinks among the blog sites, demonstrated the viability of link structure for information discovery, though their analysis of text content was less extensive. In contrast, the text seems to be of interest to social scientists studying blogs as an artifact of the political process. Although attempts to quantitatively analyze the contents of political texts have been made, results from classical, supervised text classification experiments are mixed (Mullen and Malouf, 2006; Malouf and Mullen, 2007). Also, a consensus on useful, reliable annotation or categorization schemes for political texts, at any level of granularity, has yet to emerge.

Meanwhile, latent topic modeling has become a widely used unsupervised text analysis tool. The basic aim of those models is to discover recurring patterns of "topics" within a text collection. LDA was introduced by Blei et al. (2003) and has been especially popular because it can be understood as a generative model and because it discovers understandable topics in many scenarios (Steyvers and Griffiths, 2007). Its declarative specification makes it easy to extend for new kinds of text collections. The technique has been applied to Web document collections, notably for community discovery in social networks (Zhang et al., 2007), opinion mining in user reviews (Titov and McDonald, 2008), and sentiment discovery in free-text annotations (Branavan et al., 2008). Dredze et al. (2008) applied LDA to a collection of email for summary keyword extraction. The authors evaluated the model with proxy tasks such as recipient prediction. More closely related to the data considered in this work, Lin et al. (2008) applied a variation of LDA to ideological discourse.

A notable trend in the recent research is to augment the models to describe non-textual evidence alongside the document collection. Several such studies are especially relevant to our work. Blei and Jordan (2003) were one of the earliest results in this trend. The concept was developed into more general framework by Blei and McAuliffe (2008). Steyvers et al. (2004) and Rosen-Zvi et al. (2004) first extended LDA to explicitly model the influence of *authorship*, applying the model to a collection of academic papers from CiteSeer. The model combined the ideas from the mixture model proposed by McCallum (1999) and LDA. In this model, an abstract notion "author" is associated with a distribution over topics. Another approach to the same document collection based on LDA was used for citation network analysis. Erosheva et al. (2004), following Cohn and Hofmann (2001), defined a generative process not only for each word in the text, but also its citation to other documents in the collection, thereby capturing the notion of *relations* between the document into one generative process. Nallapati and Cohen (2008) introduced the Link-PLSA-LDA model, in which the contents of the citing document and the "influences" on the document (its citations to existing literature), as well as the contents of the cited documents, are modeled together. They further applied the Link-PLSA-LDA model to a blog corpus to analyze its cross citation structure via hyperlinks.

In this work, we aim to model the data *within* blog conversations, focusing on comments left by a blog community in response to a blogger's post.

3 Political Blog Data

We discuss next the dataset used in our experiments.

3.1 Corpus

We have collected blog posts and comments from 40 blog sites focusing on American politics during the period November 2007 to October 2008, contemporaneous with the presidential elections. The discussions on these blogs focus on American politics, and many themes appear: the Democratic and Republican candidates, speculation about the results of various state contests, and various aspects of international and (more commonly) domestic politics. The sites were selected to have a variety of political leanings. From this pool we chose five blogs which accumulated a large number of posts during this period: Carpetbagger (CB),[1] Daily Kos (DK),[2] Matthew Yglesias (MY),[3] Red State (RS),[4] and Right Wing News (RWN).[5] CB and MY ceased as independent bloggers in August 2008.[6] Because

[1] http://www.thecarpetbaggerreport.com
[2] http://www.dailykos.com
[3] http://matthewyglesias.theatlantic.com
[4] http://www.redstate.com
[5] http://www.rightwingnews.com
[6] The authors of those blogs now write for larger online media, CB for Washingon Monthly at http://www.

	MY	RWN	CB	RS	DK
Time span (from 11/11/07)	–8/2/08	–10/10/08	–8/25/08	–6/26/08	–4/9/08
# training posts	1607	1052	1080	2045	2146
# words (total)	110,788	194,948	183,635	321,699	221,820
(on average per post)	(68)	(185)	(170)	(157)	(103)
# comments	56,507	34,734	34,244	59,687	425,494
(on average per post)	(35)	(33)	(31)	(29)	(198)
(unique commenters, on average)	(24)	(13)	(24)	(14)	(93)
# words in comments (total)	2,287,843	1,073,726	1,411,363	1,675,098	8,359,456
(on average per post)	(1423)	(1020)	(1306)	(819)	(3895)
(on average per comment)	(41)	(31)	(41)	(27)	(20)
Post vocabulary size	6,659	9,707	7,579	12,282	10,179
Comment vocabulary size	33,350	22,024	24,702	25,473	58,591
Size of user pool	7,341	963	5,059	2,789	16,849
# test posts	183	113	121	231	240

Table 1: Details of the blog data used in this paper.

our focus in this paper is on blog posts and their comments, we discard posts on which no one commented within six days. We also remove posts with too few words: specifically, we retain a post only if it has at least five words in the main entry, and at least five words in the comment section. All posts are represented as text only (images, hyperlinks, and other non-text contents are ignored). To standardize the texts, we remove from the text 670 commonly used stop words, non-alphabet symbols including punctuation marks, and strings consisting of only symbols and digits. We also discard infrequent words from our dataset: for each word in a post's main entry, we kept it only if it appears at least one more time in some main entry. We apply the same word pruning to the comment section as well. The corpus size and the vocabulary size of the five datasets are listed in Table 1. In addition, each user's handle is replaced with a unique integer. The dataset is available for download at `http://www.ark.cs.cmu.edu/blog-data`.

3.2 Qualitative Properties of Blogs

We believe that readers' reactions to blog posts are an integral part of blogging activity. Often comments are much more substantial and informative than the post. While circumspective articles limit themselves to allusions or oblique references, readers' comments may point to heart of the matter more boldly. Opinions are expressed more blatantly in comments. Comments may help a human (or automated) reader to understand the post more clearly when the main text is too terse, stylized, or technical.

Although the main entry and its comments are certainly related and at least partially address similar topics, they are markedly different in several ways. First of all, their vocabulary is noticeably different. Comments are more casual, conversational, and full of jargon. They are less carefully edited and therefore contain more misspellings and typographical errors. There is more diversity among comments than within the single-author post, both in style of writing and in what commenters like to talk about. Depending on the subjects covered in a blog post, different types of people are inspired to respond. We believe that analyzing a piece of text based on the reaction it causes among those who read it is a fascinating problem for NLP.

Blog *sites* are also quite distinctive from each other. Their language, discussion topics, and collective political orientations vary greatly. Their volumes also vary; multi-author sites (such as DK, RS) may consistently produce over twenty posts per day, while single-author sites (such as MY, CB) may have a day with only one post. Single author sites also tend to have a much smaller vocabulary and range of interests. The sites are also culturally different in commenting styles; some sites are full of short interjections, while others have longer, more analytical comments. On some sites, users appear to be

`washingtonmonthly.com` and MY for Think Progress at`http://yglesias.thinkprogress.org`.

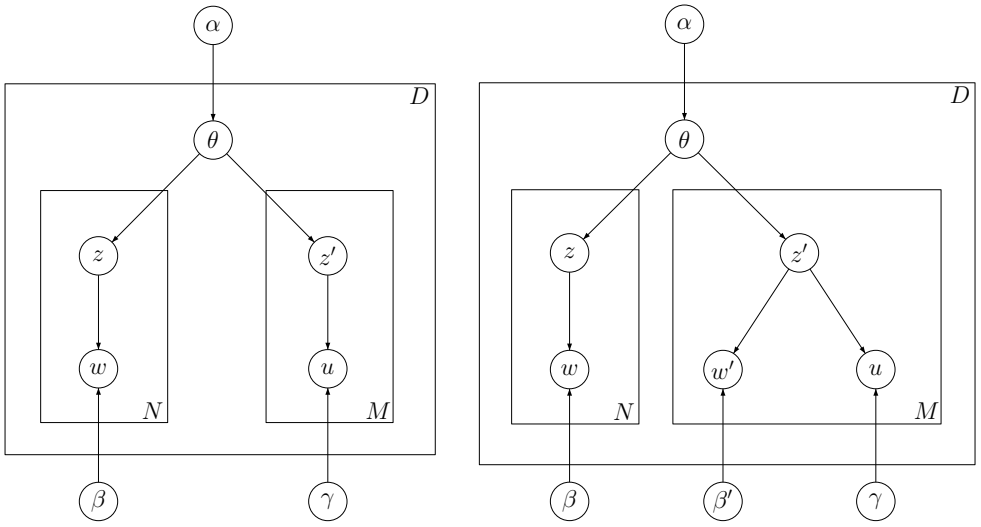


Figure 1: Left: LinkLDA (Erosheva et al., 2004), with variables reassigned. Right: CommentLDA. In training, w, u, and (in CommentLDA) w' are observed. D is the number of blog posts, and N and M are the word counts in the post and the all of its comments, respectively. Here we "count by verbosity."

close-knit, while others have high turnover.

In the next section, we describe how we apply topic models to political blogs, and how these probabilistic models can put to use to make predictions.

4 Generative Models

The first model we consider is **LinkLDA**, which is analogous to the model of Erosheva et al. (2004), though the variables are given different meanings here.[7] The graphical model is depicted in Fig. 1 (left). As in LDA and its many variants, this model postulates a set of latent "topic" variables, where each topic k corresponds to a multinomial distribution β_k over the vocabulary. In addition to generating the words in the post from its topic mixture, this model also generates a bag of users who respond to the post, according to a distribution γ over users given topics. In this model, the topic distribution θ is all that determines the text content of the post and which users will respond to the post.

LinkLDA models which users are likely to respond to a post, but it does not model what they will write. Our new model, **CommentLDA**, generates the contents of the comments (see Fig. 1, right). In order to capture the differences in language style between posts and comments, however, we use a different conditional distribution over comment words given topics, β'. The post text, comment text, and commenter distributions are all interdependent through the (latent) topic distribution θ, and a topic k is defined by:

- A multinomial distribution β_k over post words;
- A multinomial distribution β'_k over comment words; and
- A multinomial distribution γ_k over blog commenters who might react to posts on the topic.

Formally, LinkLDA and CommentLDA generate blog data as follows: For each blog post (1 to D):

1. Choose a distribution θ over topics according to Dirichlet distribution α.
2. For i from 1 to N_i (the length of the post):
   (a) Choose a topic z_i according to θ.
   (b) Choose a word w_i according to the topic's post word distribution β_{z_i}.
3. For j from 1 to M_i (the length of the comments on the post, in words):
   (a) Choose a topic z'_j.
   (b) Choose an author u_j from the topic's commenter distribution $\gamma_{z'_j}$.
   (c) *(CommentLDA only)* Choose a word w'_j according to the topic's comment word distribution $\beta'_{z'_j}$.

4.1 Variations on Counting Users

As described, CommentLDA associates each comment word token with an independent author. In both LinkLDA and CommentLDA, this "**counting by verbosity**" will force γ to give higher probability to users who write longer comments with more

[7] Instead of blog commenters, they modeled citations.

words. We consider two alternative ways to count comments, applicable to both LinkLDA and CommentLDA. These both involve a change to step 3 in the generative process.

Counting by response (replaces step 3): For j from 1 to U_i (the number of users who respond to the post): (a) and (b) as before. (c) *(CommentLDA only)* For ℓ from 1 to $\ell_{i,j}$ (the number of words in u_j's comments), choose w'_ℓ according to the topic's comment word distribution $\beta'_{z'_j}$. This model collapses all comments by a user into a single bag of words on a single topic.[8]

Counting by comments (replaces step 3): For j from 1 to C_i (the number of comments on the post): (a) and (b) as before. (c) *(CommentLDA only)* For ℓ from 1 to $\ell_{i,j}$ (the number of words in comment j), choose w'_ℓ according to the topic's comment word distribution $\beta'_{z'_j}$. Intuitively, each comment has a topic, a user, and a bag of words.

The three variations—counting users by verbosity, response, or comments—correspond to different ways of thinking about topics in political blog discourse. Counting by verbosity will let garrulous users define the topics. Counting by response is more democratic, letting every user who responds to a blog post get an equal vote in determining what the post is about, no matter how much that user says. Counting by comments gives more say to users who engage in the conversation *repeatedly*.

4.2 Implementation

We train our model using empirical Bayesian estimation. Specifically, we fix $\alpha = 0.1$, and we learn the values of word distributions β and β' and user distribution γ by maximizing the likelihood of the training data:

$$p(\boldsymbol{w}, \boldsymbol{w'}, \boldsymbol{u} \mid \alpha, \beta, \beta', \gamma) \tag{1}$$

(Obviously, β' is not present in the LinkLDA models.) This requires an inference step that marginalizes out the latent variables, θ, z, and z', for which we use Gibbs sampling as implemented by the Hierarchical Bayes Compiler (Daumé, 2007). The Gibbs sampling inference algorithm for LDA was first introduced by Griffiths and Steyvers (2004) and has since been used widely.

[8] The counting-by-response models are deficient, since they assume each user will only be chosen once per blog post, though they permit the same user to be chosen repeatedly.

5 Empirical Evaluation

We adopt a typical NLP "train-and-test" strategy that learns the model parameters on a training dataset consisting of a collection of blog posts and their commenters and comments, then considers an unseen test dataset from a later time period. Many kinds of predictions might be made about the test set and then evaluated against the true comment response. For example, the likelihood of a user to comment on the post, given knowledge of θ can be estimated as:[9]

$$\begin{aligned} p(u \mid w_1^N, \gamma, \theta) &= \sum_{z=1}^{K} p(u \mid z, \gamma) p(z \mid w_1^N, \theta) \\ &= \sum_{z=1}^{K} \gamma_{z,u} \cdot \theta_z \end{aligned} \tag{2}$$

The latter is in a sense a "guessing game," a prediction on who is going to comment on a new blog post. A similar task was used by Nallapati and Cohen (2008) for assessing the performance of Link-PLSA-LDA: they predicted the presence or absence of citation links between documents. We report the performance on this prediction task using our six blog topic models (LinkLDA and CommentLDA, with three counting variations each).

Our aim is to explore and compare the effectiveness of the different models in discovering topics that are useful for a practical task. We also give a qualitative analysis of topics learned.

5.1 Comment Prediction

For each political blog, we trained the three variations each of LinkLDA and CommentLDA. Model parameters β, γ, and (in CommentLDA) β' were learned by maximizing likelihood, with Gibbs sampling for inference, as described in §4.2. The number of topics K was fixed at 15.

A simple baseline method makes a post-independent prediction that ranks users by their comment frequency. Since blogs often have a "core constituency" of users who post frequently, this is a

[9] Another approach would attempt to integrate out θ.

	n=5	n=10	n=20	n=30	oracle
MY					
Freq.	23.93	18.68	14.20	11.65	13.18
NB	25.13	19.28	14.20	11.63	13.54
Link-v	20.10	14.04	11.17	9.23	11.32
Link-r	26.77	18.63	14.64	12.47	14.03
Link-c	25.13	18.85	14.61	11.91	13.84
Com-v	22.84	17.15	12.75	10.69	12.77
Com-r	**27.54**	**20.54**	14.61	12.45	**14.35**
Com-c	22.40	18.50	**14.83**	**12.56**	14.20
Max	*94.75*	*89.89*	*73.63*	*58.76*	*92.60*
RWN					
Freq.	32.56	30.17	22.61	19.7	**27.19**
NB	25.63	**34.86**	**27.61**	**22.03**	18.28
Link-v	28.14	21.06	17.34	14.51	19.81
Link-r	32.92	29.29	22.61	18.96	26.32
Link-c	32.56	27.43	21.15	17.43	25.09
Com-v	29.02	24.07	19.07	16.04	22.71
Com-r	**36.10**	29.64	23.8	19.26	25.97
Com-c	32.03	27.43	19.82	16.25	23.88
Max	*90.97*	*76.46*	*52.56*	*37.05*	*96.16*
CB					
Freq.	33.38	28.84	24.17	20.99	21.63
NB	36.36	31.15	25.08	**21.40**	23.22
Link-v	32.06	26.11	19.79	17.43	18.31
Link-r	**37.02**	31.65	24.62	20.85	22.34
Link-c	36.03	**32.06**	**25.28**	21.10	**23.44**
Com-v	32.39	26.36	20.95	18.26	19.85
Com-r	35.53	29.33	24.33	20.22	22.02
Com-c	33.71	29.25	23.80	19.86	21.68
Max	*99.66*	*98.34*	*88.88*	*72.53*	*95.58*
RS					
Freq.	**25.45**	16.75	11.42	9.62	17.15
NB	22.07	16.01	11.60	9.76	16.50
Link-v	14.63	11.9	9.13	7.76	11.38
Link-r	25.19	**16.92**	**12.14**	**9.82**	**17.98**
Link-c	24.50	16.45	11.49	9.32	16.76
Com-v	14.97	10.51	8.46	7.37	11.3 0
Com-r	15.93	11.42	8.37	6.89	10.97
Com-c	17.57	12.46	8.85	7.34	12.14
Max	*80.77*	*62.98*	*40.95*	*29.03*	*91.86*
DK					
Freq.	24.66	19.08	15.33	13.34	9.64
NB	**35.00**	**27.33**	**22.25**	**19.45**	**13.97**
Link-v	20.58	19.79	15.83	13.88	10.35
Link-r	33.83	27.29	21.39	19.09	13.44
Link-c	28.66	22.16	18.33	16.79	12.60
Com-v	22.16	18.00	16.54	14.45	10.92
Com-r	33.08	25.66	20.66	18.29	12.74
Com-c	26.08	20.91	17.47	15.59	11.82
Max	*100.00*	*100.00*	*100.00*	*99.09*	*98.62*

Table 2: Comment prediction results on 5 blogs. See text.

strong baseline. We also compared to a Naïve Bayes classifier (with word counts in the post's main entry as features). To perform the prediction task with our models, we took the following steps. First, we removed the comment section (both the words and the authorship information) from the test data set. Then, we ran a Gibbs sampler with the partial data, fixing the model parameters to their learned values and the blog post words to their observed values. This gives a posterior topic mixture for each post (θ in the above equations).[10] We then computed each user's comment prediction score for each post as in Eq. 2. Users are ordered by their posterior probabilities. Note that these posteriors have different meanings for different variations:

- When counting by verbosity, the value is the probability that the next (or any) comment word will be generated by the user, given the blog post.
- When counting by response, the value is the probability that the user will respond *at all*, given the blog post. (Intuitively, this approach best matches the task at hand.)
- When counting by comments, the value is the probability that the next (or any) comment will be generated by the user, given the blog post.

We compare our commenter ranking-by-likelihood with the actual commenters in the test set. We report in Tab. 2 the precision (macro-averaged across posts) of our predictions at various cut-offs (n). The oracle column is the precision where it is equal to the recall, equivalent to the situation when the true number of commenters is known. (The performance of random guessing is well below 1% for all sites at cut-off points shown.) "Freq." and "NB" refer to our baseline methods. "Link" refers to LinkLDA and "Com" to CommentLDA. The suffixes denote the counting methods: verbosity ("-v"), response ("-r"), and comments ("-c"). Recall that we considered only the comments by the users seen at least once in the training set, so perfect precision, as well as recall, is impossible when new users comment on a post; the *Max* row shows the maximum performance possible given the set of commenters recognizable from the training data.

[10] For a few cases we checked the stability of the sampler and found results varied by less than 1% precision across ten runs.

Our results suggest that, if asked to guess 5 people who would comment on a new post given some site history, we will get 25–37% of them right, depending on the site, given the content of a new post.

We achieved some improvement over both the baseline and Naïve Bayes for some cut-offs on three of the five sites, though the gains were very small for and RS and CB. LinkLDA usually works slightly better than CommentLDA, except for MY, where CommentLDA is stronger, and RS, where CommentLDA is extremely poor. Differences in commenting style are likely to blame: MY has relatively long comments in comparison to RS, as well as DK. MY is the only site where CommentLDA variations consistently outperformed LinkLDA variations, as well as Naïve Bayes classifiers. This suggests that sites with more terse comments may be too sparse to support a rich model like CommentLDA.

In general, counting by response works best, though counting by comments is a close rival in some cases. We observe that counting by response tends to help LinkLDA, which is ignorant of the word contents of the comment, more than it helps CommentLDA. Varying the counting method can bring as much as 10% performance gain.

Each of the models we have tested makes different assumptions about the behavior of commenters. Our results suggest that commenters on different sites behave differently, so that the same modeling assumptions cannot be made universally. In future work, we hope to permit blog-specific properties to be automatically discovered during learning, so that, for example, the comment words can be exploited when they are helpful but assumed independent when they are not. Of course, improved performance might also be obtained with more topics, richer priors over topic distributions, or models that take into account other cues, such as the time of the post, pages it links to, etc. It is also possible that better performance will come from more sophisticated supervised models that do not use topics.

5.2 Qualitative Evaluation

Aside from prediction tasks such as above, the model parameters by themselves can be informative. β defines which words are likely to occur in the post body for a given topic. β' tells which words are likely to appear in the collective response to a particular topic. Similarity or divergence of the two distributions can tell us about differences in language used by bloggers and their readers. γ expresses users' topic preferences. A pair or group of participants may be seen as "like-minded" if they have similar topic preferences (perhaps useful in collaborative filtering).

Following previous work on LDA and its extensions, we show words most strongly associated with a few topics, arguing that some coherent clusters have been discovered. Table 3 shows topics discovered in MY using CommentLDA (counting by comments). This is the blog site where our models most consistently outperformed the Naïve Bayes classifiers and LinkLDA, therefore we believe the model was a good fit for this dataset.

Since the site is concentrated on American politics, many of the topics look alike. Table 3 shows the most probable words in the posts, comments, and both together for five hand-picked topics that were relatively transparent. The probabilistic scores of those words are computed with the scoring method suggested by Blei and Lafferty (in press).

The model clustered words into topics pertaining to religion and domestic policy (first and last topics in Table 3) quite reasonably. Some of the religion-related words make sense in light of current affairs.[11] Some words in the comment section are slightly off-topic from the issue of religion, such as *dawkins*[12] or *wright*,[13] but are relevant in the context of real-world events. Notice those words rank highly only in the comment section, showing differences between discussion in the post and the comments. This is also noticeable, for example, in the "primary" topic (second in Table 3), where the Republican primary receives more discussion in the main post, and in the "Iraq war" and "energy" topics, where bloggers discuss strategy and commenters

[11] Mitt Romney was a candidate for the Republican nomination in 2008 presidential election. He is a member of The Church of Jesus Christ of Latter-Day Saints. Another candidate, Mike Huckabee, is an ordained Southern Baptist minister. Moktada al-Sadr is an Iraqi theologian and political activist, and John Hagee is an influential televangelist.

[12] Richard Dawkins is a well known evolutionary biologist who is a vocal critic of intelligent design.

[13] We believe this is a reference to Rev. Jeremiah Wright of Trinity United Church of Christ, whose inflammatory rhetoric was negatively associated with then-candidate Barack Obama.

religion; in both:	people, just, american, church, believe, god, black, jesus, mormon, faith, jews, right, say, mormons, religious, point
in posts:	romney, huckabee, muslim, political, hagee, cabinet, mitt, consider, true, anti, problem, course, views, life, real, speech, moral, answer, jobs, difference, muslims, hardly, going, christianity
in comments:	religion, think, know, really, christian, obama, white, wright, way, said, good, world, science, time, dawkins, human, man, things, fact, years, mean, atheists, blacks, christians
primary; in both:	obama, clinton, mccain, race, win, iowa, delegates, going, people, state, nomination, primary, hillary, election, polls, party, states, voters, campaign, michigan, just
in posts:	huckabee, wins, romney, got, percent, lead, barack, point, majority, ohio, big, victory, strong, pretty, winning, support, primaries, south, rules
in comments:	vote, think, superdelegates, democratic, candidate, pledged, delegate, independents, votes, white, democrats, really, way, caucuses, edwards, florida, supporters, wisconsin, count
Iraq war; in both:	american, iran, just, iraq, people, support, point, country, nuclear, world, power, military, really, government, war, army, right, iraqi, think
in posts:	kind, united, forces, international, presence, political, states, foreign, countries, role, need, making, course, problem, shiite, john, understand, level, idea, security, main
in comments:	israel, sadr, bush, state, way, oil, years, time, going, good, weapons, saddam, know, maliki, want, say, policy, fact, said, shia, troops
energy; in both:	people, just, tax, carbon, think, high, transit, need, live, going, want, problem, way, market, money, income, cost, density
in posts:	idea, public, pretty, course, economic, plan, making, climate, spending, economy, reduce, change, increase, policy, things, stimulus, cuts, low, fi nancial, housing, bad, real
in comments:	taxes, fuel, years, time, rail, oil, cars, car, energy, good, really, lot, point, better, prices, pay, city, know, government, price, work, technology
domestic policy; in both:	people, public, health, care, insurance, college, schools, education, higher, children, think, poor, really, just, kids, want, school, going, better
in posts:	different, things, point, fact, social, work, large, article, getting, inequality, matt, simply, percent, tend, hard, increase, huge, costs, course, policy, happen
in comments:	students, universal, high, good, way, income, money, government, class, problem, pay, americans, private, plan, american, country, immigrants, time, know, taxes, cost

Table 3: The most probable words for some CommentLDA topics (MY).

focus on the tangible (*oil*, *taxes*, *prices*, *weapons*).

While our topic-modeling approach achieves mixed results on the prediction task, we believe it holds promise as a way to understand and summarize the data. Without CommentLDA, we would not be able to easily see the differences noted above in blogger and commenter language. In future work, we plan to explore models with weaker independence assumptions among users, among blog posts over time, and even across blogs. This line of research will permit a more nuanced understanding of language in the blogosphere and in political discourse more generally.

6 Conclusion

In this paper we applied several probabilistic topic models to discourse within political blogs. We introduced a novel comment prediction task to assess these models in an objective evaluation with possible practical applications. The results show that predicting political discourse behavior is challenging, in part because of considerable variation in user behavior across different blog sites. Our results show that using topic modeling, we can begin to make reasonable predictions as well as qualitative discoveries about language in blogs.

Acknowledgments

This research was supported by a gift from Microsoft Research and NSF IIS-0836431. The authors appreciate helpful comments from the anonymous reviewers, Ja-Hui Chang, Hal Daumé, and Ramesh Nallapati. We thank Shay Cohen for his help with inference algorithms and the members of the ARK group for reviewing this paper.

References

L. Adamic and N. Glance. 2005. The political blogosphere and the 2004 U.S. election: Divided they blog. In *Proceedings of the 2nd Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*.

D. Blei and M. Jordan. 2003. Modeling annotated data. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*.

D. Blei and J. Lafferty. In press. Topic models. In A. Srivastava and M. Sahami, editors, *Text Mining: Theory and Applications*. Taylor and Franci.

D. Blei and J. McAuliffe. 2008. Supervised topic models. In *Advances in Neural Information Processing Systems 20*.

D. Blei, A. Ng, and M. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

S. R. K. Branavan, H. Chen, J. Eisenstein, and R. Barzilay. 2008. Learning document-level semantic properties from free-text annotations. In *Proceedings of ACL-08: HLT*.

D. Cohn and T. Hofmann. 2001. The missing link—a probabilistic model of document content and hypertext connectivity. In *Neural Information Processing Systems 13*.

H. Daumé. 2007. HBC: Hierarchical Bayes compiler. `http://www.cs.utah.edu/~hal/HBC`.

M. Dredze, H. M. Wallach, D. Puller, and F. Pereira. 2008. Generating summary keywords for emails using topics. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*.

E. Erosheva, S. Fienberg, and J. Lafferty. 2004. Mixed membership models of scientific publications. *Proceedings of the National Academy of Sciences*, pages 5220–5227, April.

T. L. Griffiths and M. Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101 Suppl. 1:5228–5235, April.

W.-H. Lin, E. Xing, and A. Hauptmann. 2008. A joint topic and perspective model for ideological discourse. In *Proceedings of 2008 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

R. Malouf and T. Mullen. 2007. Graph-based user classification for informal online political discourse. In *Proceedings of the 1st Workshop on Information Credibility on the Web*.

A. McCallum. 1999. Multi-label text classification with a mixture model trained by EM. In *AAAI Workshop on Text Learning*.

T. Mullen and R. Malouf. 2006. A preliminary investigation into sentiment analysis of informal political discourse. In *Proceedings of AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs*.

R. Nallapati and W. Cohen. 2008. Link-PLSA-LDA: A new unsupervised model for topics and influence of blogs. In *Proceedings of the 2nd International Conference on Weblogs and Social Media*.

M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P Smyth. 2004. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*.

M. Steyvers and T. Griffiths. 2007. Probabilistic topic models. In T. Landauer, D. Mcnamara, S. Dennis, and W. Kintsch, editors, *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum Associates.

M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. L. Griffiths. 2004. Probabilistic author-topic models for information discovery. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*.

I. Titov and R. McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL-08: HLT*.

H. Zhang, B. Qiu, C. L. Giles, H. C. Foley, and J. Yen. 2007. An LDA-based community structure discovery approach for large-scale social networks. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics*.

An Iterative Reinforcement Approach for Fine-Grained Opinion Mining

Weifu Du
Haerbin Institute of Technology
Haerbin, China
duweifu@software.ict.ac.cn

Songbo Tan
Institute of Computing Technology
Beijing, China
tansongbo@software.ict.ac.cn

Abstract

With the in-depth study of sentiment analysis research, finer-grained opinion mining, which aims to detect opinions on different review features as opposed to the whole review level, has been receiving more and more attention in the sentiment analysis research community recently. Most of existing approaches rely mainly on the template extraction to identify the explicit relatedness between product feature and opinion terms, which is insufficient to detect the implicit review features and mine the hidden sentiment association in reviews, which satisfies (1) the review features are not appear explicit in the review sentences; (2) it can be deduced by the opinion words in its context. From an information theoretic point of view, this paper proposed an iterative reinforcement framework based on the improved information bottleneck algorithm to address such problem. More specifically, the approach clusters product features and opinion words simultaneously and iteratively by fusing both their semantic information and co-occurrence information. The experimental results demonstrate that our approach outperforms the template extraction based approaches.

1 Introduction

In the Web2.0 era, the Internet turns from a static information media into a platform for dynamic information exchanging, on which people can express their views and show their selfhood. More and more people are willing to record their feelings (blog), give voice to public affairs (news review), express their likes and dislikes on products (product review), and so on. In the face of the volume of sentimental information available on the Internet continues to increase, there is growing interest in helping people better find, filter, and manage these resources.

Automatic opinion mining (Turney et al., 2003; Ku et al., 2006; Devitt et al., 2007) can play an important role in a wide variety of more flexible and dynamic information management tasks. For example, with the help of sentiment analysis system, in the field of public administration, the administrators can receive the feedbacks on one policy in a timelier manner; in the field of business, manufacturers can perform more targeted updates on products to improve the consumer experience.

The research of opinion mining began in 1997, the early research results mainly focused on the polarity of opinion words (Hatzivassiloglou et al., 1997) and treated the text-level opinion mining as a classification of either positive or negative on the number of positive or negative opinion words in one text (Turney et al., 2003; Pang et al., 2002; Zagibalov et al., 2008;). With the in-depth study of opinion mining, researchers committed their efforts for more accurate results: the research of sentiment summarization (Philip et al., 2004; Hu et al., KDD 2004), domain transfer problem of the sentiment analysis (Kanayama et al., 2006; Tan et al., 2007; Blitzer et al., 2007; Tan et al., 2008; Andreevskaia et al., 2008; Tan et al., 2009) and fine-grained opinion mining (Hatzivassiloglou et al., 2000; Takamura et al., 2007; Bloom et al., 2007; Wang et al., 2008; Titov et al., 2008) are the main branches of the research of opinion mining. In this paper, we focus on the fine-grained (feature-level) opinion mining.

For many applications (e.g. the task of public affairs review analysis and the products review analysis), simply judging the sentiment orientation of a review unit is not sufficient. Researchers (Kushal, 2003; Hu et al., KDD 2004; Hu et al., AAAI 2004; Popescu et al., 2005) began to work on finer-grained opinion mining which predicts the sentiment orientation related to different review features. The task is known as feature-level opinion mining.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 486–493,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

In feature-level opinion mining, most of the existing researches associate product features and opinion words by their explicit co-occurrence. Template extraction based method (Popescu et al., 2005) and association rule mining based method (Hu et al., AAAI 2004) are the representative ones.

These approaches did good jobs for identifying the review features that appear explicitly in reviews, however, real reviews from customers are usually complicated. In some cases, the review features are implicit in the review sentences, but can be deduced by the opinion words in its context. The detection of such hidden sentiment association is a big challenge in feature-level opinion mining on Chinese reviews due to the nature of Chinese language (Qi et al., 2008). Obviously, neither the template extraction based method nor the association rule mining based method is effective for such cases. Moreover, in some cases, even if the review features appear explicitly in the review sentences, the co-occurrence information between review features and opinion words is too quantitatively sparse to be utilized. So we consider whether it is a more sensible way to construct or cluster review feature groups and opinion words groups to mine the implicit or hidden sentiment association in the reviews.

The general approach will cluster the two types of objects separately, which neglects the highly interrelationship. To address this problem, in this paper, we propose an iterative reinforcement framework, under which we cluster product features and opinion words simultaneously and iteratively by fusing both their semantic information and sentiment link information. We take improved information bottleneck algorithm (Tishby, 1999) as the kernel of the proposed framework.

The information bottleneck approach was presented by Tishby (1999). The basic idea of the approach is that it treats the clustering problems from the information compressing point of view, and takes this problem as a case of much more fundamental problem: *what are the features of the variable X that are relevant for the prediction of another, relevance, variable Y*? Based on the information theory, the problem can be formulated as: *find a compressed representation of the variable X, denoted C, such that the mutual information between C and Y is as high as possible, under a constraint on the mutual information between X and C*. For our case, take the hotel reviews as example, *X* is one type of objects of review features (e.g. facilities, service, surrounding environment, etc) or opinion words (e.g. perfect, circumspect, quiet, etc), and *Y* is another one. Given some review features (or opinion words) gained from review corpus, we want to assemble them into categories, conserving the information about opinion words (or review features) as high as possible.

The information bottleneck algorithm has some benefits, mainly including (1) it treats the trade-off of precision versus complexity of clustering model through the rate distortion theory, which is a subfield of information theory; (2) it defines the "distance" or "similarity" in a well-defined way based on the mutual information. The efficiency of information bottleneck algorithm (Slonim and Tishby, 2000) motivates us to take it as the kernel of our framework. As far as we know, this approach has not been employed in opinion mining yet.

In traditional information bottleneck approach, the distance between two data objects is measured by the Jensen-Shannon divergence (Lin, 1991), which aims to measure the divergence between two probability distributions. We alter this measure to integrate more semantic information, which will be illustrated in detail in the following sections, and the experimental result shows the effectiveness of the alteration.

It would be worthwhile to highlight several aspects of our work here:

- We propose an iterative reinforcement framework, and under this framework, review feature words and opinion words are organized into categories in a simultaneous and iterative manner.
- In the process of clustering, the semantic information and the co-occurrence information are integrated.
- The experimental results on real Chinese web reviews demonstrate that proposed method outperforms the template extraction based algorithm.

2 Proposed Algorithm

2.1 The Problem

In product reviews, opinion words are used to express opinion, sentiment or attitude of reviewers. Although some review units may express general opinions toward a product, most review units are

regarding to specific features of the product.

A product is always reviewed under a certain feature set F. Suppose we have got a lexical list O which includes all the opinion expressions and their sentiment polarities. For the feature-level opinion mining, identifying the sentiment association between F and O is essential. The key points in the whole process are as follows:

- get opinion word set O (with polarity labels)
- get product feature set F
- identify relationships between F and O

The focus of the paper is on the latter two steps, especially for the case of hidden sentiment association that the review features are implicit in the review sentences, but can be deduced by the opinion words in its context. In contrast to existing explicit adjacency approaches, the proposed approach detects the sentiment association between F and O based on review feature categories and opinion word groups gained from the review corpus.

To this end, we first consider two sets of association objects: the set of product feature words $F = \{f_1, f_2, \ldots, f_m\}$ and the set of opinion words $O = \{o_1, o_2, \ldots o_n\}$. A weighted bipartite graph from F and O can be built, denoted by $G = \{F, O, R\}$. Here $R = [r_{ij}]$ is the $m*n$ link weight matrix containing all the pair-wise weights between set F and O. The weight can be calculated with different weighting schemes, in this paper, we set r_{ij} by the co-appearance frequency of f_i and o_j in clause level.

We take F and O as two random variables, and the question of constructing or clustering the object groups can be defined as finding compressed representation of each variable that reserves the information about another variable as high as possible. Take F as an example, we want to find its compression, denoted as C, such that the mutual information between C and O is as high as possible, under a constraint on the mutual information between F and C.

We propose an iterative reinforcement framework to deal with the tasks. An improved information bottleneck algorithm is employed in this framework, which will be illustrated in detail in the following sections.

2.2 Information Bottleneck Algorithm

The information bottleneck method (IB) was presented by Tishby et al. (1999). According to Shannon's information theory (Cover and Thomas, 1991), for the two random variables X, Y, the mutual information $I(X;Y)$ between the random variables X, Y is given by the symmetric functional:

$$I(X;Y) = \sum_{x \in X, y \in Y} p(x)\, p(y \mid x) \log \frac{p(y \mid x)}{p(y)} \qquad (1)$$

and the mutual information between them measures the relative entropy between their joint distribution $p(x, y)$ and the product of respective marginal distributions $p(x)p(y)$, which is the only consistent statistical measure of the information that variable X contains about variable Y (and vice versa). Roughly speaking, some of the mutual information will be lost in the process of compression, e.g. $I(C,Y) \le I(X,Y)$ (C is a compressed representation of X).

This representation is defined through a (possibly stochastic) mapping between each value $x \in X$ to each representative value $c \in C$. Formally, this mapping can be characterized by a conditional distribution $p(c|x)$, inducing a soft partitioning of X values, Specifically, each value of X is associated with all the codebook elements (C values), with some normalized probability.

The IB method is based on the following simple idea. Given the empirical joint distribution of two variables, one variable is compressed so that the mutual information about the other variable is preserved as much as possible. The method can be considered as finding a minimal sufficient partition or efficient relevant coding of one variable with respect to the other one. This problem can be solved by introducing a Lagrange multiplier β, and then minimizing the functional:

$$L[p(c \mid x)] = I(C,X) - \beta I(C,Y) \qquad (2)$$

This solution is given in terms of the three distributions that characterize every cluster $c \in C$, the prior probability for this cluster, $p(c)$, its membership probabilities $p(c|x)$, and its distribution over the relevance variable $p(y|c)$. In general, the membership probabilities, $p(c|x)$ is "soft", i.e. every $x \in X$ can be assigned to every $c \in C$ in some (normalized) probability. The information bottleneck principle determines the distortion measure between the points x and c to be the $D_{KL}\left[p(y \mid x) \,\|\, p(y \mid c)\right] = \sum_y p(y \mid x) \log \frac{p(y \mid x)}{p(y \mid c)}$, the Kullback-Leibler divergence (Cover and Thomas, 1991) between the conditional distributions $p(y|x)$

and $p(y|c)$. Specifically, the formal optimal solution is given by the following equations which must be solved together.

$$\begin{cases} p(c \mid x) = \dfrac{p(c)}{Z(\beta, x)} \exp(-\beta D_{KL}[\, p(y \mid x) \,\|\, p(y \mid c)]) \\ p(y \mid c) = \dfrac{1}{p(c)} \sum_x p(c \mid x)\, p(x)\, p(y \mid x) \\ p(c) = \sum_x p(c \mid x)\, p(x) \end{cases} \tag{3}$$

Where $Z(\beta, x)$ is a normalization factor, and the single positive (Lagrange) parameter β determines the "softness" of the classification. Intuitively, in this procedure the information contained in X about Y 'squeezed' through a compact 'bottleneck' of clusters C, that is forced to represent the 'relevant' part in X w.r.t to Y.

An important special case is the "hard" clustering case where C is a deterministic function of X. That is, $p(c|x)$ can only take values of zero or one, This restriction, which corresponds to the limit $\beta \to \infty$ in Eqs 3 meaning every $x \in X$ is assigned to exactly one cluster $c \in C$ with a probability of one and to all the others with a probability of zero. This yields a natural distance measure between distributions which can be easily implemented in an agglomerative hierarchical clustering procedure (Slonim and Tishby, 1999).

$$\begin{cases} p(c \mid x) = \begin{cases} 1, & \textit{if } x \in c \\ 0, & \textit{otherwise} \end{cases} \\ p(y \mid c) = \dfrac{1}{p(c)} \displaystyle\sum_{x \in c} p(x, y) \\ p(c) = \displaystyle\sum_{x \in c} p(x) \end{cases} \tag{4}$$

The algorithm starts with a trivial partitioning into $|X|$ singleton clusters, where each cluster contains exactly one element of X. At each step we merge two components of the current partition into a single new component in a way that locally minimizes the loss of mutual information about the categories. Every merger, $(c_i, c_j) \Rightarrow c_*$, is formally defined by the following equation:

$$\begin{cases} p(c_* \mid x) = \begin{cases} 1, & x \in c_i \ \textit{or}\ x \in c_j \\ 0, & \textit{otherwise} \end{cases} \\ p(y \mid c_*) = \dfrac{p(c_i)}{p(c_*)}\, p(y \mid c_i) + \dfrac{p(c_j)}{p(c_*)}\, p(y \mid c_j) \\ p(c_*) = p(c_i) + p(c_j) \end{cases} \tag{5}$$

The decrease in the mutual information $I(C, Y)$ due to this merger is defined by

$$\delta I(c_i, c_j) \equiv I(C_{before}, Y) - I(C_{after}, Y) \tag{6}$$

When $I(C_{before}, Y)$ and $I(C_{after}, Y)$ are the information values before and after the merger, respectively. After a little algebra, one can see

$$\delta I(c_i, c_j) \equiv \big(p(c_i) + p(c_j)\big) \cdot D_{JS}\big[\, p(y \mid c_i), p(y \mid c_j) \,\big] \tag{7}$$

Where the functional D_{JS} is the Jensen-Shannon divergence (Lin, 1991), defined as

$$D_{JS}\big[\, p_i, p_j \,\big] = \pi_i D_{KL}\Big[\, p_i \,\|\, \hat{p} \,\Big] + \pi_j D_{KL}\Big[\, p_j \,\|\, \hat{p} \,\Big] \tag{8}$$

where in our case

$$\begin{cases} \{\, p_i, p_j \,\} \equiv \{\, p(y \mid c_i), p(y \mid c_j) \,\} \\ \{\, \pi_i, \pi_j \,\} \equiv \left\{ \dfrac{p(c_i)}{p(c_*)}, \dfrac{p(c_j)}{p(c_*)} \right\} \\ \hat{p} = \pi_i\, p(y \mid c_i) + \pi_j\, p(y \mid c_j) \end{cases} \tag{9}$$

By introducing the information optimization criterion the resulting similarity measure directly emerges from the analysis. The algorithm is now very simple. At each step we perform "the best possible merger", i.e. merge the clusters $\{c_i, c_j\}$ which minimize $\delta I(c_i, c_j)$.

2.3 Improved Information Bottleneck Algorithm for Semantic Information

In traditional information bottleneck approach, the distance between two data objects is measured by the difference of information values before and after the merger, which is measured by the Jensen-Shannon divergence. This divergence aims to measure the divergence between two probability distributions. For our case, the divergence is based on the co-occurrence information between the two variables F and O.

While the co-occurrence in corpus is usually quantitatively sparse; additionally, Statistics based

on word-occurrence loses semantic related information. To avoid such reversed effects, in the proposed framework we combine the co-occurrence information and semantic information as the final distance between the two types of objects.

$$D(X_i, X_j) = \alpha D_{semantic}(X_i, X_j) + (1-\alpha)\delta I(X_i, X_j)$$
$$where \{X_i \in F \wedge X_j \in F\} \vee \{X_i \in O \wedge X_j \in O\} \quad (10)$$

In equation 10, the distance between two data objects X_i and X_j is denoted as a linear combination of semantic distance and information value difference. The parameter α reflects the contribution of different distances to the final distance.

Input: Joint probability distribution $p(f,o)$
Output: A partition of F into m clusters, $\forall m \in \{1,\ldots,|F|\}$, and a partition of O into n clusters $\forall n \in \{1,\ldots,|O|\}$

1. $t \leftarrow 0$
2. Repeat
   a. Construct $CF^t \leftarrow F^t$
   b. $\forall i, j=1,\ldots,|CF^t|$, $i<j$, calculate
      $d_{ij}^t \leftarrow \alpha D_{semantic}(cf_i^t, cf_j^t) + (1-\alpha)\delta I(cf_i^t, cf_j^t)$
   c. for $m \leftarrow |CF^t|$-1 to 1
      1) find the indices $\{i, j\}$, for which d_{ij}^t is minimized
      2) merge $\{cf_i^t, cf_j^t\}$ into cf_*^t
      3) update $CF^t \leftarrow \{CF^t - \{cf_i^t, cf_j^t\}\} \cup \{cf_*^t\}$
      4) update d_{ij}^t costs w.r.t cf_*^t
   d. Construct $CO^t \leftarrow O^t$
   e. $\forall i, j=1,\ldots,|CO^t|$, $i<j$, calculate
      $d_{ij}^t \leftarrow \alpha D_{semantic}(co_i^t, co_j^t) + (1-\alpha)\delta I(co_i^t, co_j^t)$
   f. for $n \leftarrow |CO^t|$-1 to 1
      1) find the indices $\{i, j\}$, for which d_{ij}^t is minimized
      2) merge $\{co_i^t, co_j^t\}$ into co_*^t
      3) update $CO^t \leftarrow \{CO^t - \{co_i^t, co_j^t\}\} \cup \{co_*^t\}$
      4) update d_{ij}^t costs w.r.t co_*^t
   g. $t \leftarrow t+1$
3. until ($CF^t = CF^{t-1}$ *and* $CO^t = CO^{t-1}$)

Figure 1: Pseudo-code of semantic information bottleneck in iterative reinforcement framework

The semantic distance can be got by the usage of lexicon, such as WordNet (Budanitsky and Hirst, 2006). In this paper, we use the Chinese lexicon HowNet[1].

The basic idea of the iterative reinforcement principle is to propagate the clustered results between different type data objects by updating their inter-relationship spaces. The clustering process can begin from an arbitrary type of data object. The clustering results of one data object type update the interrelationship thus reinforce the data object categorization of another type. The process is iterative until clustering results of both object types converge. Suppose we begin the clustering process from data objects in set F, and then the steps can be expressed as Figure 1. After the iteration, we can get the strongest n links between product feature categories and opinion word groups. That constitutes our set of sentiment association.

3 Experimental Setup

In this section we describe our experiments and the data used in these experiments.

3.1 Data

Our experiments take hotel reviews (in Chinese) as example. The corpus used in the experiments is composed by 4000 editor reviews on hotel, including 857,692 Chinese characters. They are extracted from www.ctrip.com. Each review contains a user's rating represented by "stars", the number of the star denotes the user's satisfaction. The detailed information is illustrated in Table 1,

Table 1: The detail information of corpus

User's rating	Number
1 star	555
2 star	1375
3 star	70
4 star	2000

Then we use ICTCLAS[2], a Chinese word segmentation software to extract candidate review features and opinion words.

Usually, adjectives are normally used to express opinions in reviews. Therefore, most of the existing researches take adjectives as opinion words. In the research of Hu et al. (2004), they proposed that

[1] http://www.keenage.com/

[2] www.searchforum.org.cn

other components of a sentence are unlikely to be product features except for nouns and noun phrases. Some researchers (Fujii and Ishikawa, 2006) targeted nouns, noun phrases and verb phrases. The adding of verb phrases caused the identification of more possible product features, while brought lots of noises. So in this paper, we follow the points of Hu's, extracting nouns and noun phrases as candidate product feature words.

Take the whole set of nouns and noun phrases as candidate features will bring some noise. In order to reduce such adverse effects, we use the function of Named Entity Recognition (NER) in ICTCLAS to filter out named entities, including: person, location, organization. Since the NEs have small probability of being product features, we prune the candidate nouns or noun phrases which have the above NE taggers.

Table 2: The number of candidate review features and opinion words in our corpus

Extracted Instance	Total	Non-Repeated
Candidate review feature	86,623	15,249
Opinion word	26,721	1,231

By pruning candidate product feature words, we get the set of product feature words *F*. And the set of opinion words *O* is composed by all the adjectives in reviews. The number of candidate product feature words and opinion words extracted from the corpus are shown as Table 2:

3.2 Experimental Procedure

We evaluate our approach from two perspectives:

1) Effectiveness of product feature category construction by mutual reinforcement based clustering;

2) Precision of sentiment association between product feature categories and opinion word groups;

4 Experimental Results and Discussion

4.1 Evaluation of Review Feature Category Construction

To calculate agreement between the review feature category construction results and the correct labels, we make use of the Rand index (Rand, 1971). This allows for a measure of agreement between two partitions, P_1 and P_2, of the same data set *D*. Each partition is viewed as a collection of n*(n-1)/2 pair wise decisions, where *n* is the size of *D*. For each pair of points d_i and d_j in *D*, P_i either assigns them to the same cluster or to different clusters. Let *a* be the number of decisions where d_i is in the same cluster as d_j in P_1 and in P_2. Let *b* be the number of decisions where the two instances are placed in different clusters in both partitions. Total agreement can then be calculated using

$$Rand(P_1, P_2) = \frac{a+b}{n(n-1)/2} \tag{11}$$

In our case, the parts of product feature words in the pre-constructed evaluation set are used to represent the data set *D*; a and b represent the partition agreements between the pairs of any two words in the parts and in the clustering results respectively.

In equation 10, the parameter α reflects the respective contribution of semantic information and co-occurrence information to the final distance. When $\alpha = 0$ or $\alpha = 1$, the co-occurrence information or the semantic information will be utilized alone.

In order to get the optimal combination of the two type of distance, we adjust the parameter α from 0 to 1(stepped by 0.2), and the accuracy of feature category construction with different α are shown in Figure 2:

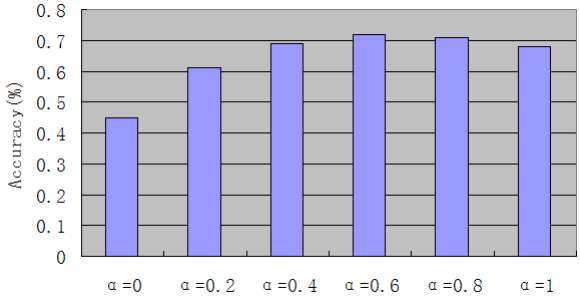


Figure 2: The accuracy of review feature category construction with the variation of the parameter α

From this figure, we can find that the semantic information (α=1) contributes much more to the accuracy of review feature category construction than the co-occurrence information (α=0), and when α=0, the approach is equivalent to the traditional information bottleneck approach. We consider this is due partly to the sparseness of the cor-

pus, by enlarging the scale of the corpus or using the search engine (e.g. google etc), we can get more accurate results.

Additionally, by a sensible adjust on the parameter α (in this experiment, we set α as 0.6), we can get higher accuracy than the two baselines (α =0 and α =1), which indicates the necessity and effectiveness of the integration of semantic information and co-occurrence information in the proposed approach.

4.2 Evaluation of Sentiment Association

We use precision to evaluate the performance of sentiment association. An evaluation set is constructed manually first, in which there are not only the categories that every review feature word belong to, but also the relationship between each category and opinion word. Then we define precision as:

$$Precision = \frac{number\ of\ correctly\ associated\ pairs}{number\ of\ detected\ pairs} \quad (12)$$

A comparative result is got by the means of template-extraction based approach on the same test set. By the usage of regular expression, the nouns (phrase) and gerund (phrase) are extracted as the review features, and the nearest adjectives are extracted as the related opinion words. Because the modifiers of adjectives in reviews also contain rich sentiment information and express the view of customs, we extract adjectives and their modifiers simultaneously, and take them as the opinion words.

Table 3: Performance comparison in sentiment association

Approach	Pairs	Precision
Template extraction	27,683	65.89%
Proposed approach	141,826	78.90%

Table 3 shows the advantage of our approach over the extraction by explicit adjacency. Using the same product feature categorization, our sentiment association approach get a more accurate pair set than the direct extraction based on explicit adjacency. The precision we obtained by the iterative reinforcement approach is 78.90%, almost 13 points higher than the adjacency approach. This indicates that there are a large number of hidden sentiment associations in the real custom reviews, which underlines the importance and value of our work.

5 Conclusions and Further Work

In this paper, we propose a novel iterative reinforcement framework based on improved information bottleneck approach to deal with the feature-level product opinion-mining problem. We alter traditional information bottleneck method by integration with semantic information, and the experimental result demonstrates the effectiveness of the alteration. The main contribution of our work mainly including:

- We propose an iterative reinforcement information bottleneck framework, and in this framework, review feature words and opinion words are organized into categories in a simultaneous and iterative manner.
- In the process of clustering, the semantic information and the co-occurrence information are integrated.
- The experimental results based on real Chinese web reviews demonstrate that our method outperforms the template extraction based algorithm.

Although our methods for candidate product feature extraction and filtering (see in 3.1) can partly identify real product features, it may lose some data and remain some noises. We'll conduct deeper research in this area in future work. Additionally, we plan to exploit more information, such as background knowledge, to improve the performance.

6 Acknowledgments

This work was mainly supported by two funds, i.e., 0704021000 and 60803085, and one another project, i.e., 2004CB318109.

References

A. Andreevskaia, S. Bergler. When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging. ACL 2008.

A. Budanitsky and G. Hirst. Evaluating wordnetbased measures of lexical semantic relatedness. Computational Linguistics, 32(1):13–47, 2006.

A. Devitt, K. Ahmad. Sentiment Polarity Identification in Financial News: A Cohesion-based Approach. ACL 2007.

A. Fujii and T. Ishikawa. A system for summarizing and visualizing arguments in subjective documents: Toward supporting decision making. The Workshop on Sentiment and Subjectivity in Text ACL2006. 2006.

A. Popescu and O. Etzioni. Extracting product features and opinions from reviews. HLT-EMNLP 2005.

B. Liu, M. Hu, and J. Cheng. Opinion observer: analyzing and comparing opinions on the web. WWW 2005.

B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. ACL 2005.

B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. EMNLP 2002.

B. Philip, T. Hastie, C. Manning, and S. Vaithyanathan. Exploring sentiment summarization. In AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications (AAAI tech report SS-04-07). 2004.

B. Wang, H. Wang. Bootstrapping Both Product Features and Opinion Words from Chinese Customer Reviews with Cross-Inducing. IJCNLP 2008.

D. Kushal, S. Lawrence, and D. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. WWW 2003.

H. Kanayama, T. Nasukawa. Fully Automatic Lexicon Expansion for Domain-oriented Sentiment Analysis. EMNLP 2006

H. Takamura, T. Inui. Extracting Semantic Orientations of Phrases from Dictionary. NAACL-HLT 2007.

I. Titov, R. McDonald. Modeling online reviews with multi-grain topic models. WWW 2008.

L. Ku, Y. Liang and H. Chen. Opinion Extraction, Summarization and Tracking in News and Blog Corpora. AAAI-CAAW 2006.

J. Blitzer, M. Dredze, F. Pereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. ACL 2007.

J. Lin. Divergence Measures Based on the Shannon Entropy. IEEE Transactions on Information theory, 37(1):145–151, 1991.

K. Bloom and N. Garg and S. Argamon. Extracting Appraisal Expressions. NAACL-HLT 2007.

M. Hu and B. Liu. Mining and summarizing customer reviews. KDD 2004.

M. Hu and B. Liu. Mining opinion features in customer reviews. AAAI 2004.

N. Slonim, N. Tishby. Agglomerative information bottleneck. NIPS 1999.

N. Slonim and N. Tishby. Document Clustering Using word Clusters via the Information Bottleneck Method. SIGIR 2000.

N. Slonim and N. Tishby. The power of word clusters for text classification. ECIR 2001.

N. Tishby, F. Pereira, W. Bialek. The information bottleneck method. 1999, arXiv: physics/0004057v1

P. Turney. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. ACL 2002.

P. Turney and M. Littman. Measuring Praise and Criticism: Inference of Semantic Orientation from Association. ACM Transactions on Information Systems, 2003,21(4): 315-346.

Q. Su, X. Xu, H. Guo, Z. Guo, X. Wu, X. Zhang, B. Swen and Z. Su. Hidden sentiment association in Chinese web opinion mining. WWW 2008.

S. Tan, G. Wu, H. Tang and X. Cheng. A novel scheme for domain-transfer problem in the context of sentiment analysis. CIKM 2007.

S. Tan, Y. Wang, G. Wu and X. Cheng. Using unlabeled data to handle domain-transfer problem of semantic detection. SAC 2008.

S. Tan, X. Cheng, Y. Wang and H. Xu. Adapting Naive Bayes to Domain Adaptation for Sentiment Analysis. ECIR 2009.

T. Cover and J. Thomas. Elements of Information Theory. John Wiley & Sons, New York, 1991.

T. Zagibalov, J. Carroll. Automatic Seed Word Selection for Unsupervised Sentiment Classification of Chinese Text. Coling 2008.

V. Hatzivassiloglou and K. McKeown. Predicting the semantic orientation of adjectives. ACL 1997.

V. Hatzivassiloglou and J. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. Coling 2000.

W. Rand. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association, 66, 846-850. 1971

For a few dollars less: Identifying review pages sans human labels

Luciano Barbosa
Dept. of Computer Science
University of Utah
Salt Lake City, UT 84112, USA.
lbarbosa@cs.utah.edu

Ravi Kumar **Bo Pang** **Andrew Tomkins**
Yahoo! Research
701 First Ave
Sunnyvale, CA 94089, USA.
{ravikumar,bopang,atomkins}@yahoo-inc.com

Abstract

We address the problem of large-scale automatic detection of online reviews *without* using any human labels. We propose an efficient method that combines two basic ideas: Building a classifier from a large number of noisy examples and using the structure of the website to enhance the performance of this classifier. Experiments suggest that our method is competitive against supervised learning methods that mandate expensive human effort.

1 Introduction

Shoppers are migrating to the web and online reviews are playing a critical role in affecting their shopping decisions, online and offline. According to two surveys published by comScore (2007) and Horrigan (2008), 81% of web users have done online research on a product at least once. Among readers of online reviews, more than 70% reported that the reviews had a significant influence on their purchases. Realizing this economic potential, search engines have been scrambling to cater to such user needs in innovative ways. For example, in response to a product-related query, a search engine might want to surface only review pages, perhaps via a "filter by" option, to the user. More ambitiously, they might want to dissect the reviews, segregate them into novice and expert judgments, distill sentiments, and present an aggregated "wisdom of the crowds" opinion to the user. Identifying review pages is the indispensable enabler to fulfill any such ambition; nonetheless, this problem does not seem to have been addressed at web scale before.

Detecting review webpages in a few, review-only websites is an easy, manually-doable task. A large fraction of the interesting review content, however, is present on pages outside such websites. This is where the task becomes challenging. Review pages might constitute a minority and can be buried in a multitude of ways among non-review pages — for instance, the movie review pages in `nytimes.com`, which are scattered among all news articles, or the product review pages in `amazon.com`, which are accessible from the product description page. An automatic and scalable method to identify reviews is thus a practical necessity for the next-generation search engines. The problem is actually more general than detecting reviews: it applies to detecting any "horizontal" category such as buying guides, forums, discussion boards, FAQs, etc.

Given the nature of these problems, it is tempting to use supervised classification. A formidable barrier is the labeling task itself since human labels need time and money. On the other hand, it is easier to generate an enormous number of low-quality labeled examples through purely automatic methods. This prompts the question: Can we do review detection by focusing just on the textual content of a large number of automatically obtained but low-quality labeled examples, perhaps also utilizing the site structure specific to each website? And how will it compare to the best supervised classification method? We address these questions in this paper.

Main contributions. We propose the first end-to-end method that can operate at web scale to efficiently detect review pages. Our method is based on using simple URL-based clues to automatically

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 494–502,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

partition a large collection of webpages into two noisy classes: One that consists mostly of review webpages and another that consists of a mixture of some review but predominantly non-review webpages (more details in Section 4.2).

We analyze the use of a naive Bayes classifier in this noisy setting and present a simple algorithm for review page classification. We further enhance the performance of this classifier by incorporating information about the structure of the website that is manifested through the URLs of the webpages. We do this by partitioning the website into clusters of webpages, where the clustering delicately balances the information in the site-unaware labels provided by the classifier in the previous step and the site structure encoded in the URL tokens; a decision tree is used to accomplish this. Our classification method for noisily-labeled examples and the use of site-specific cues to improve upon a site-independent classifier are general techniques that may be applicable in other large-scale web analyses.

Experiments on 2000 hand-labeled webpages from 40 websites of varying sizes show that besides being computationally efficient, our human-label-free method not only outperforms those based on off-the-shelf subjectivity detection but also remains competitive against the state-of-the-art supervised text classification that relies on editorial labels.

2 Related work

The related work falls into roughly four categories: Document- and sentence-level subjectivity detection, sentiment analysis in the context of reviews, learning from noisy labeled examples, and exploiting site structure for classification.

Given the subjective nature of reviews, document-level subjectivity classification is closely related to our work. There have been a number of approaches proposed to address document-level subjectivity in news articles, weblogs, etc. (Yu and Hatzivassiloglou, 2003; Wiebe et al., 2004; Finn and Kushmerick, 2006; Ni et al., 2007; Stepinski and Mittal, 2007). Ng et al. (2006) experiment with review identification for known domains using datasets with clean labels (e.g., movie reviews vs. movie-related non-reviews), a setting different from that of ours. Pang and Lee (2008b) present a method on re-ranking documents that are web search results for a specific query (containing the word `review`) based on the subjective/objective distinction. Given the nature of the query, they implicitly detect reviews from unknown sources. But their re-ranking algorithm only applies to webpages known to be (roughly) related to the same narrow subject. Since the webpages in our datasets cover not only a diverse range of websites but also a diverse range of topics, their approach does not apply. To the best of our knowledge, there has been no work on identifying review pages at the scale and diversity we consider.

Subjectivity classification of within-document items, such as terms, has been an active line of research (Wiebe et al. (2004) present a survey). Identifying subjective sentences in a document via off-the-shelf packages is an alternative way of detecting reviews without (additional) human annotations. In particular, the OpinionFinder system (Riloff and Wiebe, 2003; Wiebe and Riloff, 2005) is a state-of-the-art knowledge-rich sentiment-analysis system. We will use it as one of our baselines and compare its performance with our methods.

There has been a great deal of previous work in sentiment analysis that worked with reviews, but they were typically restricted to using reviews extracted from one or two well-known sources, bypassing automatic review detection. Examples of such early work include (Turney, 2002; Pang et al., 2002; Dave et al., 2003; Hu and Liu, 2004; Popescu and Etzioni, 2005). See Pang and Lee (2008a) for a more comprehensive survey. Building a collection of diverse review webpages, not limited to one or two hosts, can better facilitate such research.

Learning from noisy examples has been studied for a long time in the learning theory community (Angluin and Laird, 1988). Learning naive Bayes classifiers from noisy data (either features or labels or both) was studied by Yang et al. (2003). Their focus, however, is to reconstruct the underlying conditional probability distributions from the observed noisy dataset. We, on the other hand, rely on the volume of labels to drown the noise. Along this spirit, Snow et al. (2008) show that obtaining multiple low-quality labels (through Mechanical Turk) can approach high-quality editorial labels. Unlike in their setting, we do not have multiple low-quality labels for the same URL. The extensive body of work in

semi-supervised learning or learning from one class is also somewhat relevant to our work. A major difference is that they tend to work with small amount of clean, labeled data. In addition, many semi-supervised/transductive learning algorithms are not efficient for web-scale data.

Using site structure for web analysis tasks has been addressed in a variety of contexts. For example, Kening et al. (2005) exploit the structure of a website to improve classification. On a related note, co-training has also been used to utilize inter-page link information in addition to intra-page textual content: Blum and Mitchell (1998) use anchor texts pointing to a webpage as the alternative "view" of the page in the context of webpage classification. Their algorithm is largely site-unaware in that it does not explicitly exploit site structures. Utilizing site structures also has remote connections to wrapper induction, and there is extensive literature on this topic. Unfortunately, the methods in all of these work require human labeling, which is precisely what our work is trying to circumvent.

3 Methodology

In this section we describe our basic methodology for identifying review pages. Our method consists of two main steps. The first is to use a large amount of noisy training examples to learn a basic classifier for review webpages; we adapt a simple naive Bayes classifier for this purpose. The second is to improve the performance of this basic classifier by exploiting the website structure; we use a decision tree for this.

Let P be the set of all webpages. Let C_+ denote the *positive* class, i.e., the set of all review pages and let C_- denote the *negative* class, i.e., the set of all non-review pages. Each webpage p is exactly in one of C_+ or C_-, and is labeled $+1$ or -1 respectively.

3.1 Learning from large amounts of noisy data

Previous work using supervised or semi-supervised learning approaches for sentiment analysis assumes relatively high-quality labels that are produced either via human annotation or automatically generated through highly accurate rules (e.g., assigning positive or negative label to a review according to automatically extracted star ratings).

We examine a different scenario where we can automatically generate large amount of relatively low-quality labels. Section 4.2 describes the process in more detail, but briefly, in a collection of pages crawled from sites that are very likely to host reviews, those with the word `review` in their URLs are very likely to contain reviews (the noisy positive set $\widetilde{C}_+$) and the rest of the pages on those sites are less likely to contain reviews (the more noisy negative set $\widetilde{C}_-$). More formally, for a webpage p, suppose $\Pr[p \in C_+ \mid p \in \widetilde{C}_+] = \alpha$ and $\Pr[p \in C_+ \mid p \in \widetilde{C}_-] = \beta$, where $1 > \alpha \gtrsim \beta > 0$. Can we still learn something useful from $\widetilde{C}_+$ and $\widetilde{C}_-$ despite the labels being highly noisy?

The following analysis is based on a naive Bayes classifier. We chose naive Bayes classifier since the learning phase can easily be parallelized.

Given a webpage (or a document) p represented as a bag of features $\{f_i\}$, we wish to assign a class $\arg\max_{c \in \{C_+, C_-\}} \Pr[c \mid p]$ to this webpage. Naive Bayes classifiers assume f_i's to be conditionally independent and we have $\Pr[p \mid c] = \prod \Pr[f_i \mid c]$. Let $r_i = \Pr[f_i \mid C_+] / \Pr[f_i \mid C_-]$ denote the contribution of each feature towards classification, and $\mathrm{rc} = \Pr[C_+] / \Pr[C_-]$ denote the ratio of class priors. First note that

$$\log \frac{\Pr[C_+|p]}{\Pr[C_-|p]} = \log \left(\frac{\Pr[C_+]}{\Pr[C_-]} \cdot \frac{\Pr[p|C_+]}{\Pr[p|C_-]} \right)$$
$$= \log \left(\frac{\Pr[C_+]}{\Pr[C_-]} \cdot \prod r_i \right) = \log \mathrm{rc} + \sum \log r_i.$$

A webpage p receives label $+1$ iff $\Pr[C_+ \mid p] > \Pr[C_- \mid p]$, and by above, if and only if $\sum \log r_i > -\log \mathrm{rc}$.

When we do not have a reasonable estimate of $\Pr[C_+]$ and $\Pr[C_-]$, as in our setting, the best we can do is to assume $\mathrm{rc} = 1$. In this case, p receives label $+1$ if and only if $\sum \log r_i > 0$. Thus, a feature f_i with $\log r_i > 0$ has a positive contribution towards p being labeled $+1$; call f_i to be a "positive" feature. Typically we use relative-frequency estimation of $\Pr[c]$ and $\Pr[f_i \mid c]$ for $c \in \{C_+, C_-\}$. Now, how does the estimation from a dataset with noisy labels compare with the estimation from a dataset with clean labels?

To examine this, we calculate the following:

$$\Pr[f_i \mid \widetilde{C}_+] = \alpha \Pr[f_i \mid C_+] + (1 - \alpha) \Pr[f_i \mid C_-],$$
$$\Pr[f_i \mid \widetilde{C}_-] = \beta \Pr[f_i \mid C_+] + (1 - \beta) \Pr[f_i \mid C_-].$$

Let $\tilde{r}_i = \frac{\Pr[f_i|\widetilde{C}_+]}{\Pr[f_i|\widetilde{C}_-]} = \frac{\alpha r_i + (1-\alpha)}{\beta r_i + (1-\beta)}$. Clearly $\tilde{r}_i$ is monotonic but not linear in r_i. Furthermore, it is bounded:

$(1-\alpha)/(1-\beta) \leq \tilde{r}_i \leq \alpha/\beta$. However,

$\tilde{r}_i > 1 \iff \alpha r_i + (1-\alpha) > \beta r_i + (1-\beta)$

$\iff (\alpha-\beta) r_i > (\alpha-\beta) \iff r_i > 1$, where the last step used $\alpha > \beta$. Thus, the sign of $\log \tilde{r}_i$ is the same as that of $\log r_i$, i.e., a feature contributing positively to $\sum \log r_i$ will continue to contribute positively to $\sum \log \tilde{r}_i$ (although its magnitude is distorted) and vice versa.

The above analysis motivates an alternative model to naive Bayes. Instead of each feature f_i placing a weighted vote $\log \tilde{r}_i$ in the final decision, we trust only the sign of $\log \tilde{r}_i$, and let each feature f_i place a vote for the class C_+ (respectively, C_-) if $\log \tilde{r}_i > 0$ (respectively, $\log \tilde{r}_i < 0$). Intuitively, this model just compares the number of "positive" features and the number of "negative" features, ignoring the magnitude (since it is distorted anyway). This is precisely our algorithm: For a given threshold γ, the final label $nbu_\gamma(p)$ of a webpage p is given by

$$nbu_\gamma(p) = \text{sgn}\left(\sum \text{sgn}(\log \tilde{r}_i) - \gamma\right),$$

where sgn is the sign function. For comparison purposes, we also indicate the "weighted" version:

$$nbw_\gamma(p) = \text{sgn}\left(\sum \log \tilde{r}_i - \gamma\right).$$

If $\gamma = 0$, we omit γ and use nb to denote a generic label assigned by any of the above algorithms.

Note that even though our discussions were for two-class and in particular, review classification, they are equally applicable to a wide range of classification tasks in large-scale web-content analysis. Our analysis of learning from automatically generated noisy examples is thus of independent interest.

3.2 Utilizing site structure

Can the structure of a website be exploited to improve the classification of webpages given by $nb(\cdot)$? While not all websites are well-organized, quite a number of them exhibit certain structure that makes it possible to identify large subsites that contain only review pages. Typically but not always this structure is manifested through the tokens in the URL corresponding to the webpage. For instance, the pattern `http://www.zagat.com/verticals/PropertyDetails.aspx?VID=a&R=b`, where `a,b` are numbers, is indicative of all webpages in `zagat.com` that are reviews of restaurants. In fact, we can think of this as a generalization of having the keyword `review` in the URL. Now, suppose we have an initial labeling $nb(p) \in \{\pm 1\}$ for each webpage p produced by a classifier (as in the previous section, or one that is trained on a small set of human annotated pages), can we further improve the labeling using the pattern in the URL structure?

It is not immediate how to best use the URL structure to identify the review subsites. First, URLs contain irrelevant information (e.g., the token `verticals` in the above example), thus clustering by simple cosine similarity may not discover the review subsites. Second, the subsite may not correspond to a subtree in the URL hierarchy, i.e., it is not reasonable to expect all the review URLs to share a common prefix. Third, the URLs contain a mixture of path components (e.g., `www.zagat.com/verticals/PropertyDetails.aspx`) and key-value pairs (e.g., `VID=a` and `R=b`) and hence each token (regardless of its position) in the URL could play a role in determining the review subsite. Furthermore, conjunction of presence/absence of certain tokens in the URL may best correspond to subsite membership. In light of these, we represent each URL (and hence the corresponding webpage) by a bag $\{g_i\}$ of tokens obtained from the URL. We perform a crude form of feature selection by dropping tokens that are either ubiquitous (occurring in more than 99% of URLs) or infrequent (occurring in fewer than 1% of URLs) in a website; neither yields useful information.

Our overall approach will be to use g_i's to partition P into clusters $\{C_i\}$ of webpages such that each cluster C_i is predominantly labeled as either review or non-review by $nb(\cdot)$. This automatically yields a new label $cls(p)$ for each page p, which is the majority label of the cluster of p:

$$cls(p) = \text{sgn}\left(\sum_{q \in C(p)} nb(q)\right),$$

where $C(p)$ is the cluster of p. To this end, we use a decision tree classifier to build the clusters. This classifier will use the features $\{g_i\}$ and the target labels $nb(\cdot)$. The classifier is trained on all the webpages in the website and in the obtained decision tree, each leaf, consisting of pages with the same set of feature values leading down the path, corresponds to a cluster of webpages. Note that the clusters delicately balance the information in the site-unaware labels $nb(\cdot)$ and the site structure encoded

in the URLs (given by g_i's). Thus the label $\mathit{cls}(p)$ can be thought of as a *smoothed* version of $\mathit{nb}(p)$.

Even though we can expect most clusters to be homogeneous (i.e., pure reviews or non-reviews), the above method can produce clusters that are inherently heterogeneous. This can happen if the website URLs are organized such that many subsites contain both review and non-review webpages. To take this into account, we propose the following hybrid approach that interpolates between the unsmoothed labels given by $\mathit{nb}(\cdot)$ and the smoothed labels given by $\mathit{cls}(\cdot)$. For a cluster C_i, the *discrepancy* $\mathrm{disc}(C_i) = \sum_{p \in C_i} [\mathit{cls}(p) \neq \mathit{nb}(p)]$; this quantity measures the number of disagreements between the majority label $\mathit{cls}(p)$ and the original label $\mathit{nb}(p)$ for each page p in the cluster. The decision tree guarantees $\mathrm{disc}(C_i) \leq |C_i|/2$. We call a cluster C_i to be δ-*homogeneous* if $\mathrm{disc}(C_i) \leq \delta |C_i|$, where $\delta \in [0, 1/2]$. For a fixed δ, the hybrid label of a webpage p is given by

$$\mathit{hyb}_\delta(p) = \begin{cases} \mathit{cls}(p) & \text{if } C(p) \text{ is } \delta\text{-homogeneous,} \\ \mathit{nb}(p) & \text{otherwise.} \end{cases}$$

Note that $\mathit{hyb}_{1/2}(p) = \mathit{cls}(p)$ and $\mathit{hyb}_0(p) = \mathit{nb}(p)$.

Note that in the above discussions, any clustering method that can incorporate the site-unaware labels $\mathit{nb}(\cdot)$ and the site-specific tokens in g_i's could have been used; off-the-shelf decision tree was merely a specific way to realize this.

4 Data

It is crucial for this study to create a dataset that is representative of a diverse range of websites that host reviews over different topics in different styles. We are not aware of any extensive index of online review websites and we do not want to restrict our study to a few well-known review aggregation websites (such as `yelp.com` or `zagat.com`) since this will not represent the less popular and more specialized ones. Instead, we utilized user-generated tags for webpages, available on social bookmarking websites such as `del.icio.us`.

We obtained (a sample of) a snapshot of URL–tag pairs from `del.icio.us`. We took the top one thousand sites with `review*` tags; these websites hopefully represent a broad coverage. We were able to crawl over nine hundred of these sites and the resulting collection of webpages served as the basis of the experiments in this paper. We refer to these websites (or the webpages from these sites, when it is clear from the context) as S_{all}.

4.1 Gold-standard test set

When the websites are as diverse as represented in S_{all}, there is no perfect automatic way to generate the ground truth labels. Thus we sampled a number of pages for human labeling as follows.

First, we set aside 40 sites as the test sites (S_{40}). In order to represent different types of websites (to the best we can), we sampled the 40 sites so that S_{40} covers different size ranges, since large-scale websites and small-scale websites are often quite different in style, topic, and content. We uniformly sampled 10 sites from each of the four size categories (roughly, sites with 100–5K, 5K–25K, 25K–100K, and 100K+ webpages)[1]. Indeed, S_{40} (as did S_{all}) covered a wide range of topics (e.g., games, books, restaurants, movies, music, and electronics) and styles (e.g., dedicated review sites, product sites that include user reviews, newspapers with movie review sections, religious sites hosting book reviews, and non-English review sites).

We then sampled 50 pages to be labeled from each site in S_{40}. Since there are some fairly large sites that have only a small number of review pages, a uniform sampling may yield no review webpages from those sites. To reflect the natural distribution on a website and to represent pages from both classes, the webpages were sampled in the following way. For each website in S_{40}, 25 pages were uniformly sampled (representing the natural distribution) and 25 pages were sampled from among "equivalence classes" based on URLs so that pages from each major URL pattern were represented. Here, each webpage in the site is represented by a URL signature containing the most frequent tokens that occur in the URLs in that site and all pages with the same signature form an equivalence class.

For our purposes, a webpage is considered a review if it contains significant amount of textual information expressing subjective opinions on or personal experiences with a given product / service. When in doubt, the guiding principle is whether

[1] As we do not want to waste human annotation on sites with no reviews at all, a quick pre-screening process eliminated candidate sites that did not seem to host any reviews.

a page can be a satisfactory result page for users searching for reviews. More specifically, the human annotation labeled each webpage, after thoroughly examining the content, with one of the following seven intuitive labels: "single" (contains exactly one review), "multiple" (concatenation of more than one review), "no" (clearly not a review page), "empty" (looks like a page that could contain reviews but had none), "login" (a valid user login needed to look at the content), "hub" (a pointer to one or more review pages), and "ambiguous" (border-line case, e.g., a webpage with a one line review). The first two labels were treated as $+1$ (i.e., reviews) and the last five labels were treated as -1 (i.e., non-reviews). Out of the 2000 pages, we obtained 578 pages labeled $+1$ and the 1422 pages labeled -1. On a pilot study using two human judges, we obtained 78% inter-judge agreement for the seven labels and 92% inter-judge agreement if we collapse the labels to ± 1. Percentages of reviews in our samples from different sites range from 14.6% to 93.9%.

Preprocessing for text-based analysis. We processed the crawled webpages using `lynx` to extract the text content. To discard templated content, which is an annoying issue in large-scale web processing, and HTML artifacts, we used the following preprocessing. First, the HTML tags `<p>`, `<br>`, `</tr>`, and `</td>` were interpreted as paragraph breaks, the '.' inside a paragraph was interpreted as a sentence break, and whitespace was used to tokenize words in a sentence. A sentence is considered "good" if it has at least seven alphabetic words and a paragraph is considered "good" if it has at least two good sentences. After extracting the text using `lynx`, only the good paragraphs were retained. This effectively removes most of the templated content (e.g., navigational phrases) and retains most of the "natural language" texts. Because of this preprocessing, 485 pages out of 2000 turned out to be empty and these were discarded (human labels on 97% of these empty pages were -1).

4.2 Dataset with noisy labels

As discussed in Section 3.1, our goal is to obtain a large noisy set of positive and negative labeled examples. We obtained these labels for the webpages in the training sites, S_{rest}, which is essentially $S_{\text{all}} \setminus S_{40}$. First, the URLs in S_{rest} were tokenized using a unigram model based on an English dictionary; this is so that strings such as `reviewoftheday` are properly interpreted.

$\widetilde{C}_+$: To be labeled $+1$, the path-component of the URL of the webpage has to contain the token `review`. Our assumption is that such pages are highly likely to be review pages. On a uniform sample of 100 such pages in S_{all}, 90% were found to be genuine reviews. Thus, we obtained a collection of webpages with slightly noisy positive labels.

$\widetilde{C}_-$: The rest of the pages in S_{rest} were labeled -1. Clearly this is a noisy negative set since not all pages containing reviews have `review` as part of their URLs (recall the example from `zagat.com`); thus many pages in $\widetilde{C}_-$ can still be reviews.

While the negative labels in S_{rest} are more noisy than the positive labels, we believe most of the non-review pages are in $\widetilde{C}_-$, and as most websites contain a significant number of non-review pages, the percentage of reviews in $\widetilde{C}_-$ is smaller than that in $\widetilde{C}_+$ (the assumption $\alpha \gneq \beta$ in Section 3.1).

We collected all the paragraphs (as defined earlier) from both $\widetilde{C}_+$ and $\widetilde{C}_-$ separately. We eliminated duplicate paragraphs (this further mitigates the templates issue, especially for sites generated by content-management software), and trained a unigram language model as in Section 3.1.

5 Evaluations

The evaluations were conducted on the 1515 labeled (non-empty) pages in S_{40} described in Section 4.1. We report the accuracy (acc.) as well as precision (prec.), recall (rec.), and f-measure (fmeas.) for C_+.

Trivial baselines. Out of the 1515 labeled pages, 565 were labeled $+1$ and 950 were labeled -1. Table 1 summarizes the performance of baselines that always predict one of the classes and a baseline that randomly select a class according to the class distribution S_{40}. As we can see, the best accuracy is .63, the best f-measure is .54, and they cannot be achieved by the same baseline. Before present-

	acc.	prec.	rec.	fmeas.
always C_-	**.63**	-	0	-
always C_+	.37	.37	1	**.54**
random	.53	.37	.37	.37

Table 1: Trivial baseline performances.

ing the main results of our methods, we introduce a much stronger baseline that utilizes a knowledge-rich subjectivity detection package.

5.1 Using subjectivity detectors

This baseline is motivated by the fact that reviews often contain extensive subjective content. There are many existing techniques that detect subjectivity in text. OpinionFinder (`http://www.cs.pitt.edu/mpqa/opinionfinderrelease/`) is a well-known system that processes documents and automatically identifies subjective sentences in them. OpinionFinder uses two subjective sentence classifiers (Riloff and Wiebe, 2003; Wiebe and Riloff, 2005). The first (denoted opf_{A}) focuses on yielding the highest accuracy; the second (denoted opf_{B}) optimizes precision at the expense of recall. The methods underlying OpinionFinder incorporate extensive tools from linguistics (including, speech activity verbs, psychological verbs, FrameNet verbs and adjectives with frame "experiencer", among others) and machine learning. In terms of performance, previous work has shown that OpinionFinder is a challenging system to improve upon for review retrieval (Pang and Lee, 2008b). Computationally, OpinionFinder is very expensive and hence unattractive for large-scale webpage analysis (running OpinionFinder on 1515 pages took about five hours). Therefore, we also propose a light-weight subjectivity detection mechanism called *lwd*, which counts the number of opinion words in each sentence in the text. The opinion words (5403 of them) were obtained from an existing subjectivity lexicon (`http://www.cs.pitt.edu/mpqa`).

We ran both opf_{A} and opf_{B} on the tokenized text (running them on raw HTML produced worse results). Each sentence in the text was labeled subjective or objective. We experimented with two ways to label a document using sentence-level subjectivity labels. We labeled a document $+1$ if it contained at least k subjective sentences (denoted as $opf_{\star}(k)$, where $k > 0$ is the absolute threshold), or at least f fraction of its sentences were labeled subjective (denoted as $opf_{\star}(f)$, where $f \in (0, 1]$ is the relative threshold). We conducted exhaustive parameter search with both opf_{A} and opf_{B}. For instance, the performances of opf_{A} as a function of the thresholds, both absolute and relative, is shown in Figure 1. Table 2 summarizes the best performances of $opf_{\star}(k)$ (first two rows) and $opf_{\star}(f)$ (next two rows), in terms of accuracy and f-measure (boldfaced). Similarly, for *lwd*, we labeled a document $+1$ if at least k sentences have at least ℓ opinion words (denoted $lwd(k, \ell)$.) Table 2 once again shows the best performing parameters for both accuracy and f-measure for *lwd*. Our results indicate that a simple method such as *lwd* can come very close to a sophisticated system such as $opf_{\star}$.

	acc.	prec.	rec.	fmeas.
$opf_{\mathrm{A}}(2)$	**.704**	.597	.634	.615
$opf_{\mathrm{B}}(2)$	.659	.526	.857	**.652**
$opf_{\mathrm{A}}(.17)$	**.652**	.529	.614	.568
$opf_{\mathrm{B}}(.36)$	.636	.523	.797	**.632**
$lwd(1, 4)$	**.716**	.631	.572	.600
$lwd(1, 1)$	.666	.538	.740	**.623**

Table 2: Best performances of $opf_{\star}$ and *lwd* methods.

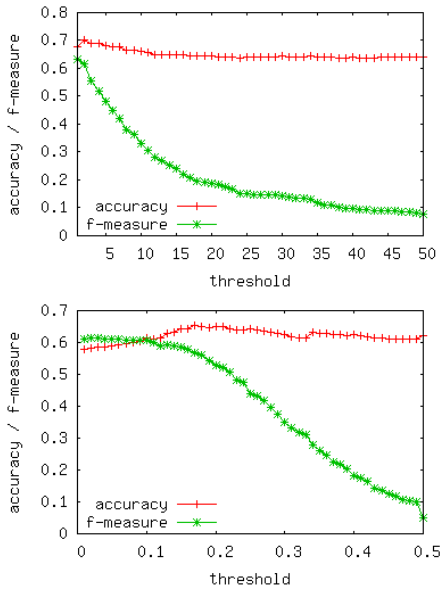


Figure 1: Performance of opf_{A} as a function of thresholds: Absolute and relative.

5.2 Main results

As stated earlier, we do not have any prior knowledge about the value of γ and hence have to work with $\gamma = 0$. To investigate the implications of this assumption, we study the performance of nbu_{γ} and nbw_{γ} as a function of γ. The accuracy and f-measures are plotted in Figure 2. There are three

	acc.	prec.	rec.	fmeas.
nbu	.753	.652	.726	.687
cls	.756	.696	.616	.654
$hyb_{1/3}$	**.777**	.712	.674	**.693**

Table 3: Performance of our methods.

conclusions that can be drawn from this study: (i) The peak values of accuracy and f-measure are comparable for both nbu_γ and nbw_γ, (ii) at $\gamma = 0$, *nbu* is much better than *nbw*, in terms of both accuracy and f-measure, and (iii) the best performance of nbu_γ occurs at $\gamma \approx 0$. Given the difficulty of obtaining γ if one were to use nbw_γ, the above conclusions validate our intuition and the algorithm in Section 3.1.

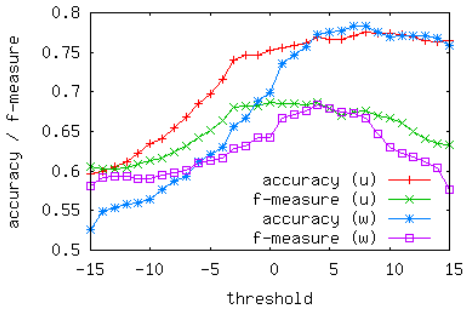


Figure 2: Performance as threshold changes: Comparing nbu_γ (marked as (u)) with nbw_γ (marked as (w)).

Table 3 shows the performance of the site-specific method outlined in Section 3.2. The clusters were generated using the unpruned J48 decision tree in Weka (`www.cs.waikato.ac.nz/ml/weka`). In our experiments, we set $\delta = 1/3$ as a natural choice for the hybrid method. As we see the performance of *nbu* is about 7% better than the best performance using a subjectivity-based method (in terms of accuracy). The performance of the smoothed labels (decision tree-based clustering) is comparable to that of *nbu*. However, the hybrid method $hyb_{1/3}$ yields an additional 3% relative improvement over *nbu*. Paired t-test over the accuracies for these 40 sites shows both $hyb_{1/3}$ and *nbu* to be statistically significantly better than the $opf_\star$ with best accuracy (with $p < 0.05$, $p < 0.005$, respectively), and $hyb_{1/3}$ to be statistically significantly better than *nbu* (with $p < 0.05$).

5.3 Cross-validation on S_{40}

While the main focus of our paper is to study how to detect reviews without human labels, we present cross validation results on S_{40} as a comparison point. The goal of this experiment is to get a sense of the best possible accuracy and f-measure numbers using labeled data and the state-of-the-art method for text classification, namely, SVMs. In other words, the performance numbers obtained through SVMs and cross-validation can be thought of as realistic "upper bounds" on the performance of content-based review detection. We used SVM^{light} (`svmlight.joachims.org`) for this purpose.

The cross-validation experiment was conducted as follows. We split the data by site to simulate the more realistic setting where pages in the test set do not necessarily come from a known site. Each fold consisted of one site from each size category; thus, 36 of the 40 sites in S_{40} were used for training and the remainder for testing. Over ten folds, the average performance was: accuracy .795, precision .759, recall .658, and f-measure .705.

Thus our methods in Section 3 come reasonably close to the "upper bound" given by SVMs and human-labeled data. In fact, while the supervised SVMs statistically significantly outperform *nbu*, they are statistically indistinguishable from $hyb_{1/3}$ via paired t-test over site-level accuracies.

6 Conclusions

In this paper we proposed an automatic method to perform efficient and large-scale detection of reviews. Our method is based on two principles: Building a classifier from a large number of noisy labeled examples and using the site structure to improve the performance of this classifier. Extensive experiments suggest that our method is competitive against supervised learning methods that depend on expensive human labels. There are several interesting avenues for future research, including improving the current method for exploiting the site structure. On a separate note, previous research has explicitly studied sentiment analysis as an application of transfer learning (Blitzer et al., 2007). Given the diverse range of topics present in our dataset, addressing topic-dependency is also an interesting future research direction.

References

Dana Angluin and Philip D. Laird. 1988. Learning from noisy examples. *Machine Learning*, 2(4):343–370.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of 45th ACL*.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of 11th COLT*, pages 92–100.

comScore. 2007. Online consumer-generated reviews have significant impact on offline purchase behavior. Press Release, November. `http://www.comscore.com/press/release.asp?press=1928`.

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of 12th WWW*, pages 519–528.

Aidan Finn and Nicholas Kushmerick. 2006. Learning to classify documents according to genre. *JASIST*, 7(5):1506–1518.

John A. Horrigan. 2008. Online shopping. Pew Internet & American Life Project Report.

Minqing Hu and Bing Liu. 2004. Mining opinion features in customer reviews. In *Proceedings of 19th AAAI*, pages 755–760.

Gao Kening, Yang Leiming, Zhang Bin, Chai Qiaozi, and Ma Anxiang. 2005. Automatic classification of web information based on site structure. In *Cyberworlds*, pages 552–558.

Vincent Ng, Sajib Dasgupta, and S. M. Niaz Arifin. 2006. Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of 21st COLING/44th ACL Poster*, pages 611–618.

Xiaochuan Ni, Gui-Rong Xue, Xiao Ling, Yong Yu, and Qiang Yang. 2007. Exploring in the weblog space by detecting informative and affective articles. In *Proceedings of 16th WWW*, pages 281–290.

Bo Pang and Lillian Lee. 2008a. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Bo Pang and Lillian Lee. 2008b. Using very simple statistics for review search: An exploration. In *Proceedings of 22nd COLING*. Poster.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP*, pages 339–346.

Ellen Riloff and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In *Proceedings of EMNLP*, pages 105–112.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast - but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*.

Adam Stepinski and Vibhu Mittal. 2007. A fact/opinion classifier for news articles. In *Proceedings of 30th SIGIR*, pages 807–808.

Peter Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of 40th ACL*, pages 417–424.

Janyce M. Wiebe and Ellen Riloff. 2005. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of CICLing*, pages 486–497.

Janyce M. Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. 2004. Learning subjective language. *Computational Linguistics*, 30(3):277–308.

Yirong Yang, Yi Xia, Yun Chi, and Richard R. Muntz. 2003. Learning naive Bayes classifier from noisy data. Technical Report 56, UCLA.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of EMNLP*, pages 129–136.

More than Words: Syntactic Packaging and Implicit Sentiment

Stephan Greene*
ATG, Inc.
1111 19th St, NW Suite 600
Washington, DC 20036
sgreene@atg.com

Philip Resnik
Linguistics / UMIACS CLIP Laboratory
University of Maryland
College Park, MD 20742
resnik@umiacs.umd.edu

Abstract

Work on sentiment analysis often focuses on the words and phrases that people use in overtly opinionated text. In this paper, we introduce a new approach to the problem that focuses not on lexical indicators, but on the syntactic "packaging" of ideas, which is well suited to investigating the identification of *implicit sentiment*, or perspective. We establish a strong predictive connection between linguistically well motivated features and implicit sentiment, and then show how computational approximations of these features can be used to improve on existing state-of-the-art sentiment classification results.

1 Introduction

As Pang and Lee (2008) observe, the last several years have seen a "land rush" in research on sentiment analysis and opinion mining, with a frequent emphasis on the identification of opinions in evaluative text such as movie or product reviews. However, sentiment also may be carried implicitly by statements that are not only non-evaluative, but not even visibly subjective. Consider, for example, the following two descriptions of the same (invented) event:

1(a) On November 25, a soldier veered his jeep into a crowded market and killed three civilians.
(b) On November 25, a soldier's jeep veered into a crowded market, causing three civilian deaths.

Both descriptions appear on the surface to be objective statements, and they use nearly the same words. Lexically, the sentences' first clauses differ only in the difference between *'s* and *his* to express the relationship between the soldier and the jeep, and in the second clauses both *kill* and *death* are terms with negative connotations, at least according to the General Inquirer lexicon (Stone, 1966). Yet the descriptions clearly differ in the feelings they evoke: if the soldier were being tried for his role in what happened on November 25, surely the prosecutor would be more likely to say (1a) to the jury, and the defense attorney (1b), rather than the reverse.[1]

Why, then, should a description like (1a) be perceived as less sympathetic to the soldier than (1b)? If the difference is not in the words, it must be in the way they are put together; that is, the *structure* of the sentence. In Section 2, we offer a specific hypothesis about the connection between structure and implicit sentiment: we suggest that the relationship is mediated by a set of "grammatically relevant" semantic properties well known to be important cross-linguistically in characterizing the interface between syntax and lexical semantics. In Section 3, we validate this hypothesis by means of a human ratings study, showing that these properties are highly predictive of human sentiment ratings. In Section 4, we introduce *observable proxies for underlying semantics* (OPUS), a practical way to approximate the relevant semantic properties automatically as features in a supervised learning setting. In Section 5, we show that these features improve on the existing state of the art in automatic sentiment classification. Sec-

*This work was done while the first author was a student in the Department of Linguistics, University of Maryland.

[1] We refer readers not sharing this intuition to Section 3.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 503–511,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

tions 6 and 7 discuss related work and summarize.

2 Linguistic Motivation

Verbal descriptions of an event often carry along with them an underlying attitude toward what is being described. By framing the same event in different ways, speakers or authors "select some aspects of a perceived reality and make them more salient in a communicating text, in such a way as to promote a particular problem definition, causal interpretation, moral evaluation, and/or treatment recommendation" (Entman, 1993, p. 52). Clearly lexical choices can accomplish this kind of selection, e.g. choosing to describe a person as a *terrorist* rather than a *freedom fighter*, or referencing *killer whales* rather than *orcas*.[2] Syntactic choices can also have framing effects. For example, Ronald Reagan's famous use of the passive construction, "Mistakes were made" (in the context of the Iran-Contra scandal), is a classic example of framing or spin: used without a *by*-phrase, the passive avoids identifying a causal agent and therefore sidesteps the issue of responsibility (Broder, 2007). A toddler who says "My toy broke" instead of "I broke my toy" is employing the same linguistic strategy.

Linguists have long studied syntactic variation in descriptions of the same event, often under the general heading of syntactic diathesis alternations (Levin, 1993; Levin and Hovav, 2005). This line of research has established a set of semantic properties that are widely viewed as "grammatically relevant" in the sense that they enable generalizations about syntactic "packaging" of meaning within (and across) the world's languages. For example, the verb *break* in English participates in the causative-inchoative alternation (causative event *X broke Y* can also be expressed without overt causation as *Y broke*), but the verb *climb* does not (*X* also causes the event in *X climbed Y*, but that event cannot be expressed as *Y climbed*). These facts about participation in the alternation turn out to be connected with the fact that a breaking event entails a *change of state* in *Y* but a climbing event does not. Grammatically relevant semantic properties of events and their participants — causation, change of state, and others — are central not only in theoretical work on lexical semantics, but in computational approaches to the lexicon, as well (e.g. (Pustejovsky, 1991; Dorr, 1993; Wu and Palmer, 1994; Dang et al., 1998)).

The approach we propose draws on two influential discussions about grammatically relevant semantic properties in theoretical work on lexical semantics. First, Dowty (1991) characterizes grammatically relevant properties of a verb's arguments (e.g. subject and object) via inferences that follow from the meaning of the verb. For example, expressions like *X murders Y* or *X interrogates Y* entail that subject X caused the event.[3] Second, Hopper and Thompson (1980) characterize "semantic transitivity" using similar properties, connecting semantic features to morphosyntactic behavior across a wide variety of languages.

Bringing together Dowty with Hopper and Thompson, we find 13 semantic properties organized into three groups, corresponding to the three components of a canonical transitive clause, expressed as *X verb Y* in English.[4] *Properties associated with X* involve volitional involvement in the event or state, causation of the event, sentience/awareness and/or perception, causing a change of state in *Y*, kinesis or movement, and existence independent of the event. *Properties associated with the event or state conveyed by the verb* include aspectual features of telicity (a defined endpoint) and punctuality (the latter of which may be inversely related to a property known as incremental theme). *Properties associated with Y* include affectedness, change of state, (lack of) kinesis or movement, and (lack of) existence independent of the event.

Now, observe that this set of semantic properties involves many of the questions that would naturally help to shape one's opinion about the event described by *veer* in (1). Was anyone or anything affected by what took place, and to what degree? Did the event just happen or was it caused? Did the event reach a defined endpoint? Did participation in

[2] Supporters of an endangered species listing in Puget Sound generally referred to the animals as *orcas*, while opponents generally said *killer whales* (Harden, 2006).

[3] Kako (2006) has verified that people make these inferences based on X's syntactic position even when a semantically empty nonsense verb is used.

[4] We are deliberately sidestepping the choice of terminology for X and Y, e.g. proto-Patient, theme, etc.

the event involve conscious thought or intent? Our hypothesis is that the syntactic aspects of "framing", as characterized by Entman, involve manipulation of these semantic properties, even when overt opinions are not being expressed. That is, we propose a connection between syntactic choices and implicit sentiment mediated by the very same semantic properties that linguists have already identified as central when connecting surface expression to underlying meaning more generally.

3 Empirical Validation

We validated the hypothesized connection between implicit sentiment and grammatically relevant semantic properties using psycholinguistic methods, by varying the syntactic form of event descriptions, and showing that the semantic properties of descriptions do indeed predict perceived sentiment.[5]

3.1 Semantic property ratings

Materials. Stimuli were constructed using 11 verbs of killing, which are widely viewed as prototypical for the semantic properties of interest here (Lemmens, 1998): *X killed Y* normally involves conscious, intentional causation by X of a kinetic event that causes a (rather decisive and clearly terminated!) change of state in Y. The verbs comprise two classes: the "transitive" class, involving externally caused change-of-state verbs (*kill*, *slaughter*, *assassinate*, *shoot*, *poison*), and the "ergative" class (*strangle*, *smother*, *choke*, *drown*, *suffocate*, *starve*), within which verbs are internally caused (McKoon and MacFarland, 2000) or otherwise emphasize properties of the object. Variation of syntactic description involved two forms: a transitive syntactic frame with a human agent as subject ("transitive form", 2a), and a nominalization of the verb as subject and the verb *kill* as the predicate ("nominalized form", 2b).

2(a) The gunmen shot the opposition leader
(b) The shooting killed the opposition leader

Participants and procedure. A set of 18 volunteer participants, all native speakers of English, were presented with event descriptions and asked to answer questions probing both Dowty's proto-role properties as well as Hopper and Thompson's semantic transitivity components, responding via ratings on a 1-to-7 scale. For example, the questions probing volition were: "*In this event*, how likely is it that ⟨subject⟩ chose to be involved?", where ⟨subject⟩ was *the gunmen* and *the shooting*, for 2(a-b), respectively.[6]

3.2 Sentiment ratings

Materials. We used the materials above to construct short, newspaper-like paragraphs, each one accompanied by a "headline" version of the same syntactic descriptions used above. For example, given this paragraph:

> A man has been charged for the suffocation of a woman early Tuesday morning. City police say the man suffocated the 24-year-old woman using a plastic garbage bag. The woman, who police say had a previous relationship with her attacker, was on her way to work when the incident happened. Based on information provided by neighbors, police were able to identify the suspect, who was arrested at gunpoint later the same day.

the three alternative headlines would be:

3(a) Man suffocates 24-year old woman
(b) Suffocation kills 24-year-old woman
(c) 24-year-old woman is suffocated

Some paragraphs were based on actual news stories.[7] In all paragraphs, there is an obvious nominal referent for both the perpetrator and the victim, it is clear that the victim dies, and the perpetrator in the scenario is responsible for the resulting death directly rather than indirectly (e.g. through negli-

[5] Full details and materials in Greene (2007).

[6] Standard experimental design methods were followed with respect to counterbalancing, block design, and distractor stimuli; for example, no participant saw more than one of 2(a) or 2(b), and all participants saw equal numbers of transitive and nominalized descriptions. The phrase *In this event* was repeated in each question and emphasized visually in order to encourage participants to focus on the particular event described in the sentence, rather than on the entities or events denoted in general.

[7] In those cases no proper names were used, to avoid any inadvertent emotional reactions or legal issues, although the descriptions retained emotional impact because we wanted readers to have some emotional basis with which to judge the headlines.

gence).[8] The stem of the nominalization always appeared in the event description in either verbal or nominal form.

Participants and procedure. A set of 31 volunteers, all native speakers of English, were presented with the paragraph-length descriptions and accompanying headlines. As a measure of sentiment, participants were asked to rate headlines on a 1-to-7 scale with respect to how sympathetic they perceive the headline to be toward the perpetrator. For example, given the paragraph and one of the associated headlines in (3), a participant would be asked to rate "How sympathetic or unsympathetic is *this headline* to the man?"[9]

3.3 Analysis and discussion

Unsurprisingly, but reassuringly, an analysis of the sentiment ratings yields a significant effect of syntactic form on sympathy toward the perpetrator ($F(2, 369) = 33.902, p < .001$), using a mixed model ANOVA run with the headline form as fixed effect. The transitive form of the headline yielded significantly lower sympathy ratings than the nominalized or passive forms in pairwise comparisons (both $p < .001$). We have thus confirmed empirically that Reagan's "Mistakes were made" was a wise choice of phrasing on his part.

More important, we are now in a position to examine the relationship between syntactic forms and perceived sentiment in more detail. We performed regression analyses treating the 13 semantic property ratings plus the identity of the verb as independent variables to predict sympathy rating as a dependent variable, using the 24 stimulus sentences that bridged both collections of ratings.[10] Considering semantic properties individually, we find that volition has the strongest correlation with sympathy (a negative correlation, with $r = -.776$), followed by sentience ($r = -.764$) and kinesis/movement ($r = -.751$). Although performing a multiple regression with all variables for this size dataset is impossible, owing to overfitting (as a rule of thumb, 5 to 10 observed items are necessary per each independent variable), a multiple regression involving verb, volition, and telicity as independent variables yields $R = .88$, $R^2 = .78$ ($p < .001$). The value for adjusted R^2, which explicitly takes into account the small number of observations, is 74.1.

In summary, then, this ratings study confirms the influence of syntactic choices on perceptions of implicit sentiment. Furthermore, it provides support for the idea that this influence is mediated by "grammatically relevant" semantic properties, demonstrating that these accounted for approximately 75% of the variance in implicit sentiment expressed by alternative headlines describing the same event.

4 Observable Approximation

Thus far, we have established a predictive connection between syntactic choices and underlying or implicit sentiment, mediated by grammatically relevant semantic properties. In an ideal world, we could harness the predictive power of those properties by using volition, causation, telicity, etc. as features for regression or classification in sentiment prediction tasks. Unfortunately, the properties are not directly observable, and neither automatic annotators nor labeled training data currently exist.

We therefore pursue a different strategy, which we refer to as *observable proxies for underlying semantics* (OPUS). It can be viewed as a middle ground between relying on construction-level syntactic distinctions (such as the 3-way transitive, nominalized subject, passive distinction in Section 3) and annotation of fine-grained semantic properties. The key idea is to use observable grammatical relations, drawn from the usages of terms determined to be relevant to a domain, as proxies for the underlying semantic properties that gave rise to their syntactic realization using those relations. Automatically cre-

[8] An alert reader may observe that headlines with nominalized subjects using the verb *kill* require some other nominalization, so they don't say "Killing kills victim". For these cases in the data, an appropriate nominalization drawn from the event description was used (e.g., *explosion*).

[9] Again, standard experimental design methods were used with respect to block design, distractor stimuli, etc. The phrase *this headline* was emphasized to stress that it is the headline being rated, not the story. A second question rating sympathy toward the victim was also asked in each case, as an additional distractor.

[10] These involved only the transitive and nominalized forms, because many of the questions were inapplicable to the passive form. Since the two ratings studies involved different subject pools, regression models were run over the mean values of each observation in the experimental data.

ated features based on those observable proxies are then used in classification as described in Section 5.

In order to identify the set T of terms relevant to a particular document collection, we adopt the relative frequency ratio (Damerau, 1993), $R(t) = R^t_{\text{domain}}/R^t_{\text{reference}}$, where $R^t_c = \frac{f^t_c}{N_c}$ is the ratio of term t's frequency in corpus c to the size N_c of that corpus. $R(t)$ is a simple but effective comparison of a term's prevalence in a particular collection as compared to a general reference corpus. We used the British National Corpus as the reference because it is both very large and representative of text from a wide variety of domains and genres. The threshold of $R(t)$ permitting membership in T is an experimental parameter.

OPUS features are defined in terms of syntactic dependency relations involving terms in T. Given a set D of syntactic dependency relations, features are of the form $t : d$ or $d : t$, with $d \in D, t \in T$. That is, they are term-dependency pairs extracted from term-dependency-term dependency tuples, preserving whether the term is the head or the dependent in the dependency relation. In addition, we add two construction-specific features: TRANS:v, which represents verb v in a canonical, syntactically transitive usage, and NOOBJ:v, present when verb v is used without a direct object.[11]

Example 4 shows source text (bolded clause in 4a), an illustrative subset of parser dependencies (4b), and corresponding OPUS features (4c):

4(a) Life Without Parole does not eliminate the risk that **the prisoner will murder a guard, a visitor, or another inmate**.

(b) nsubj(murder, prisoner); aux(murder, will); dobj(murder, guard)

(c) TRANS:murder, murder:nsubj, nsubj:prisoner, murder:aux, aux:will, murder:dobj, dobj:guard

Intuitively the presence of TRANS:murder suggests the entire complex of semantic properties discussed in Section 2, bringing together the impliciation of volition, causation, etc. on the part of *prisoner* (as does nsubj:prisoner), affectedness and change of state on the part of *guard* (as does dobj:guard), and so forth.

[11]We parsed English text using the Stanford parser.

The NOOBJ features can capture a habitual reading, or in some cases a detransitivizing effect associated with omission of the direct object (Olsen and Resnik, 1997). The bold text in (5) yields NOOBJ:kill as a feature.

5(a) At the same time, we should never ignore the risks of allowing the inmate **to kill again**.

In this case, omitting the direct object decreases the extent to which the killing event is interpreted as telic, and it eliminates the possibility of attributing change-of-state to a specific affected object (much like "Mistakes were made" avoids attributing cause to a specified subject), placing the phrasing at a less "semantically transitive" point on the transitivity continuum (Hopper and Thompson, 1980). Some informants find a perceptible increase in negative sentiment toward *inmate* when the sentence is phrased as in 5(b):

5(b) At the same time, we should never ignore the risks of allowing the inmate **to kill someone again**.

5 Computational Application

Having discussed linguistic motivation, empirical validation, and practical approximation of semantically relevant features, we now present two studies demonstrating their value in sentiment classification. For the first study, we have constructed a new data set particularly well suited for testing our approach, based on writing about the death penalty. In our second study, we make a direct comparison with prior state-of-the-art classification using the Bitter Lemons corpus of Lin et al. (2006).

5.1 Predicting Opinions of the Death Penalty

Corpus. We constructed a new corpus for experimentation on implicit sentiment by downloading the contents of pro- and anti-death-penalty Web sites and manually checking, for a large subset, that the viewpoints expressed in documents were as expected. The collection, which we will refer to as the DP corpus, comprises documents from five pro-death-penalty sites and three anti-death-penalty sites, and the corpus was engineered to have an even balance, 596 documents per side.[12]

[12]Details in Greene (2007).

Frequent bigram baseline. We adopted a supervised classification approach based on word n-gram features, using SVM classification in the WEKA machine learning package. In initial exploration using both unigrams and bigrams, and using both word forms and stems, we found that performance did not differ significantly, and chose stemmed bigrams for our baseline comparisons. In order to control for the difference in the number of features available to the classifier in our comparisons, we use the N most frequent stemmed bigrams as the baseline feature set where N is matched to number of OPUS features used in the comparison condition.

OPUS-kill verbs: OPUS features for manually selected verbs. We created OPUS features for 14 verbs — those used in Section 3, plus *murder*, *execute*, and *stab* and their nominalizations (including both event and *-er* nominals, e.g. both *killing* and *killer*) — generating $N = 1016$ distinct features.

OPUS-domain: OPUS features for domain-relevant verbs. We created OPUS features for the 117 verbs for which the relative frequency ratio was greater than 1. This list includes many of the *kill* verbs we used in Section 3, and introduces, among others, many transitive verbs describing acts of physical force (e.g. *rape*, *rob*, *steal*, *beat*, *strike*, *force*, *fight*) as well as domain-relevant verbs such as *testify*, *convict*, and *sentence*. Included verbs near the borderline included, for example, *hold*, *watch*, *allow*, and *try*. Extracting OPUS features for these verbs yielded $N = 7552$ features.

Evaluation. Cross-validation at the document level does not test what we are interested in, since a classifier might well learn to bucket documents according to Web site, not according to pro- or anti-death-penalty sentiment. To avoid this difficulty, we performed site-wise cross-validation. We restricted our attention to the two sites from each perspective with the most documents, which we refer to as *pro1*, *pro2*, *anti1*, and *anti2*, yielding 4-fold cross-validation. Each fold $f_{train,test}$ is defined as containing all documents from one pro and one anti site for training, using all documents from the remaining pro and anti sites for testing. So, for example, fold $f_{11,22}$ uses all documents from *pro1* and *anti1* in training, and all documents from *pro2* and *anti2* for testing.[13] As Table 1 shows, OPUS features provide substantial and statistically significant gains ($p < .001$).

Condition	N features	**SVM accuracy**
Baseline	1016	68.37
OPUS-kill verbs	1016	82.09
Baseline	7552	71.96
OPUS-domain	7552	88.10

Table 1: Results for 4-fold site-wise cross-validation using the DP corpus

Condition	N features	**SVM accuracy**
Baseline	1518	55.95
OPUS-frequent verbs	1518	55.95
OPUS-kill verbs	1062	66.67

Table 2: DP corpus comparison for OPUS features based on frequent vs. domain-relevant verbs

As a reality check to verify that it is domain-relevant verb usages and the encoding of events they embody that truly drives improved classification, we extracted OPUS features for the 14 most frequent verbs found in the DP Corpus that were *not* in our manually created list of kill verbs, along with their nominalizations. Table 2 shows the results of a classification experiment using a single train-test split, training on 1062 documents from *pro1*, *pro2*, *anti1*, *anti2* and testing on 84 test documents from the significantly smaller remaining sites. Using OPUS features for the most frequent non-*kill* verbs fails to beat the baseline, establishing that it is not simply term frequency, the presence of particular grammatical relations, or a larger feature set that the *kill*-verb OPUS model was able to exploit, but rather the properties of event encodings involving the *kill* verbs themselves.

5.2 Predicting Points of View in the Israeli-Palestinian Conflict

In order to make a direct comparison here with prior state-of-the-art work on sentiment analysis, we report on sentiment classification using OPUS features in experiments using a publicly available corpus involving opposing perspectives, the Bitter Lemons

[13] Site (# of documents): pro1= clarkprosecutor.org (437), pro2= prodeathpenalty.com (117), anti1= deathpenaltyinfo.org (319), anti2= nodeathpenalty.org (212)

(hence BL) corpus introduced by Lin et al. (2006).

Corpus. The Bitter Lemons corpus comprises essays posted at `www.bitterlemons.org`, which, in the words of the site, "present Israeli and Palestinian viewpoints on prominent issues of concern". As a corpus, it has a number of interesting properties. First, its topic area is one of significant interest and considerable controversy, yet the general tenor of the web site is one that eschews an overly shrill or extreme style of writing. Second, the site is organized in terms of issue-focused weekly editions that include essays with contrasting viewpoints from the site's two editors, plus two essays, also contrasting, from guest editors. This creates a natural balance between the two sides and across the subtopics being discussed. The BL corpus as prepared by Lin et al. contains 297 documents from each of the Israeli and Palestinian viewpoints, averaging 700-800 words in length.

Lin et al. classifiers. Lin et al. report results on distinguishing Israeli vs. Palestinian perspectives using an SVM classifier, a naive Bayes classifier NB-M using maximum a posteriori estimation, and a naive Bayes classifier NB-B using full Bayesian inference. (Document perspectives are labeled clearly on the site.) We continue to use the WEKA SVM classifier, but compare our results to both their SVM and NB-B, since the latter achieved their best results.

OPUS features. As in Section 5.1, we experimented with OPUS features driven by automatically extracted lists of domain-relevant verbs. For these experiments, we included domain-relevant nouns, and we varied a threshold ρ for the relative frequency ratio, including only terms for which $\log(R(t)) > \rho$. In addition, we introduced a general filter on OPUS features, eliminating syntactic dependency types that do not usefully reflect semantically relevant properties: det, predet, preconj, prt, aux, auxpas, cc, punct, complm, mark, rel, ref, expl.

Evaluation. Lin et al. describe two test scenarios. In the first, referred to as Test Scenario 1, they trained on documents written by the site's guests, and tested on documents from the site's editors. Test Scenario 2 represents the reverse, training on documents from the site editors and testing on documents from guest authors. As in our site-wise cross validation for the DP corpus, this strategy ensures that what is being tested is classification according to the viewpoint, not author or topic.

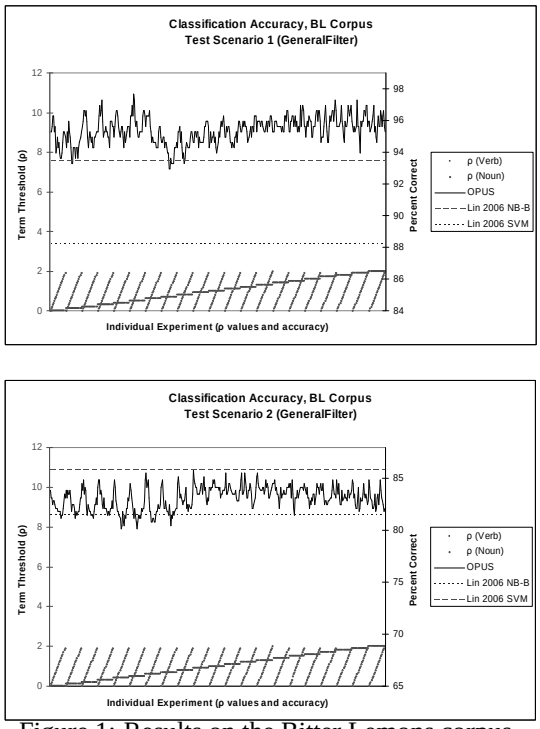


Figure 1: Results on the Bitter Lemons corpus

Figure 1 (top) summarizes a large set of experiments for Test Scenario 1, in which we varied the values of ρ for verbs and nouns. Each experiment, using a particular $\langle \rho(\text{verbs}), \rho(\text{nouns}) \rangle$, corresponds to a vertical strip on the x-axis. The points on that strip include the ρ values for verbs and nouns, measured by the scale on the y-axis at the left of the figure; the accuracy of Lin et al.'s SVM (88.22% accuracy, constant across all our variations); the accuracy of Lin et al.'s NB-B classifier (93.46% accuracy, constant across all our variations), and the accuracy of our SVM classifier using OPUS features, which varies depending on the ρ values. Across 423 experiments, our average accuracy is 95.41%, with the best accuracy achieved being 97.64%. Our classifier underperformed NB-B slightly, with accuracies from 92.93% to 93.27%, in just 8 of the 423 experiments.

Figure 1 (bottom) provides a similar summary for

experiments in Test Scenario 2. The first thing to notice is that accuracy for all methods is lower than for Test Scenario 1. This is not terribly surprising: it is likely that training a classifier on the more uniform authorship of the editor documents builds a model that generalizes less well to the more diverse authorship of the guest documents (though accuracy is still quite high). In addition, the editor-authored documents comprise a smaller training set, consisting of 7,899 sentences, while the guest documents have a total of 11,033 sentences, a 28% difference. In scenario 2, we obtain average accuracy across experiments of 83.12%, with a maximum of 85.86%, in this case outperforming the 81.48% obtained by Lin's SVM fairly consistently, and in some cases approaching or matching NB-B at 85.85%.

6 Related Work

Pang and Lee's (2008) excellent monograph provides a thorough, well organized, and relatively recent description of computational work on sentiment, opinion, and subjectivity analysis.

The problem of classifying underlying sentiment in statements that are not overtly subjective is less studied within the NLP literature, but it has received some attention in other fields. These include, for example, research on content analysis in journalism, media studies, and political economy (Gentzkow and Shapiro, 2006a; Gentzkow and Shapiro, 2006b; Groseclose and Milyo, 2005; Fader et al., 2007); automatic identification of customer attitudes for business e-mail routing (Durbin et al., 2003). And, of course, the study of perceptions in politics and media bears a strong family resemblance to real-world marketing problems involving reputation management and business intelligence (Glance et al., 2005).

Within computational linguistics, what we call implicit sentiment was introduced as a topic of study by Lin et al. (2006) under the rubric of identifying perspective, though similar work had begun earlier in the realm of political science (e.g. (Laver et al., 2003)). Other recent work focusing on the notion of perspective or ideology has been reported by Martin and Vanberg (2008) and Mullen and Malouf (2008).

Among prior authors, Gamon's (2004) research is perhaps closest to the work described here, in that he uses some features based on a sentence's logical form, generated using a proprietary system. However, his features are templatic in nature in that they do not couple specific lexical entries with their logical form. Hearst (1992) and Mulder et al. (2004) describe systems that make use of argument structure features coupled with lexical information, though neither provides implementation details or experimental results.

In terms of computational experimentation, work by Thomas et al. (2006), predicting yes and no votes in corpus of United States Congressional floor debate speeches, is quite relevant. They combined SVM classification with a min-cut model on graphs in order to exploit both direct textual evidence and constraints suggested by the structure of Congressional debates, e.g. the fact that the same individual rarely gives one speech in favor of a bill and another opposing it. We have extend their method to use OPUS features in the SVM and obtained significant improvements over their classification accuracy (Greene, 2007; Greene and Resnik, in preparation).

7 Conclusions

In this paper we have introduced an approach to implicit sentiment motivated by theoretical work in lexical semantics, presenting evidence for the role of semantic properties in human sentiment judgments. This research is, to our knowledge, the first to draw an explicit and empirically supported connection between theoretically motivated work in lexical semantics and readers' perception of sentiment. In addition, we have reported positive sentiment classification results within a standard supervised learning setting, employing a practical first approximation to those semantic properties, including positive results in a direct comparison with the previous state of the art.

Because we computed OPUS features for opinionated as well as non-evaluative language in our corpora, obtaining overall positive results, we believe these features may also improve conventional opinion labeling for subjective text. This will be investigated in future work.

Acknowledgments

The authors gratefully acknowledge useful discussions with Don Hindle and Chip Denman.

References

John Broder. 2007. Familiar fallback for officials: 'mistakes were made'. *New York Times*. March 14.

F. J. Damerau. 1993. Generating and evaluating domain-oriented multi-word terms from texts. *Information Processing and Management*, 29:433–447.

Hoa Trang Dang, Karin Kipper, Martha Palmer, and Joseph Rosenzweig. 1998. Investigating Regular Sense Extensions Based on Intersective Levin Classes. In *ACL/COLING 98*, pages 293–299, Montreal, Canada, August 10–14.

Bonnie J. Dorr. 1993. *Machine Translation: A View from the Lexicon*. The MIT Press, Cambridge, MA.

David Dowty. 1991. Thematic Proto-Roles and Argument Selection. *Language*, 67:547–619.

S. D. Durbin, J. N. Richter, and D. Warner. 2003. A system for affective rating of texts. In *Proc. 3rd Workshop on Operational Text Classification, KDD-2003*.

Robert M. Entman. 1993. Framing: Toward clarification of a fractured paradigm. *Journal of Communication*, 43(4):51–58.

Anthony Fader, Dragomir R. Radev, Michael H. Crespin, Burt L. Monroe, Kevin M. Quinn, and Michael Colaresi. 2007. MavenRank: Identifying influential members of the US Senate using lexical centrality. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Michael Gamon. 2004. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proc. COLING*.

M. Gentzkow and J. Shapiro. 2006a. Media bias and reputation. *Journal of Political Economy*, 114:280–316.

M. Gentzkow and J. Shapiro. 2006b. What drives media slant? Evidence from U.S. newspapers. http://ssrn.com/abstract=947640.

Natalie Glance, Matthew Hurst, Kamal Nigam, Matthew Siegler, Robert Stockton, and Takashi Tomokiyo. 2005. Deriving marketing intelligence from online discussion. In *Proc. KDD'05*, pages 419–428, New York, NY, USA. ACM.

Stephan Greene. 2007. *Spin: Lexical Semantics, Transitivity, and the Identification of Implicit Sentiment*. Ph.D. thesis, University of Maryland.

T. Groseclose and J. Milyo. 2005. A measure of media bias. *The Quarterly Journal of Economics*, 120:1191–1237.

B. Harden. 2006. On Puget Sound, It's Orca vs. Inc. *The Washington Post*. July 26, page A3.

Marti Hearst. 1992. Direction-based text interpretation as an information access refinement. In Paul Jacobs, editor, *Text-Based Intelligent Systems*, pages 257–274. Lawrence Erlbaum Associates.

Paul Hopper and Sandra Thompson. 1980. Transitivity in Grammar and Discourse. *Language*, 56:251–295.

E. Kako. 2006. Thematic role properties of subjects and objects. *Cognition*, 101(1):1–42, August.

Michael Laver, Kenneth Benoit, and John Garry. 2003. Extracting policy positions from political texts using words as data. *American Political Science Review*, 97(2):311–331.

M. Lemmens. 1998. *Lexical perspectives on transitivity and ergativity*. John Benjamins.

Beth Levin and Malka Rappaport Hovav. 2005. *Argument Realization*. Research Surveys in Linguistics. Cambridge University Press, New York.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, IL.

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. 2006. Which side are you on? identifying perspectives at the document and sentence levels. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Lanny W. Martin and Georg Vanberg. 2008. A robust transformation procedure for interpreting political text. *Political Analysis*, 16(1):93–100.

G. McKoon and T. MacFarland. 2000. Externally and internally caused change of state verbs. *Language*, pages 833–858.

M. Mulder, A. Nijholt, M. den Uyl, and P. Terpstra. 2004. A lexical grammatical implementation of affect. In *Proc. TSD-04, Lecture notes in computer science 3206*, pages 171–178). Springer-Verlag.

Tony Mullen and Robert Malouf. 2008. Taking sides: User classification for informal online political discourse. *Internet Research*, 18:177–190.

Mari Broman Olsen and Philip Resnik. 1997. Implicit Object Constructions and the (In)transitivity Continuum. In *33rd Proceedings of the Chicago Linguistic Society*, pages 327–336.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

James Pustejovsky. 1991. The Generative Lexicon. *Computational Linguistics*, 17(4):409–441.

Philip J. Stone. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. The MIT Press.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proc. EMNLP*, pages 327–335.

Zhibao Wu and Martha Palmer. 1994. Verb Semantics and Lexical Selection. In *Proc. ACL*, pages 133–138, Las Cruces, New Mexico.

Streaming for large scale NLP: Language Modeling

Amit Goyal, Hal Daumé III, and Suresh Venkatasubramanian
University of Utah, School of Computing
{amitg,hal,suresh}@cs.utah.edu

Abstract

In this paper, we explore a streaming algorithm paradigm to handle large amounts of data for NLP problems. We present an efficient low-memory method for constructing high-order approximate n-gram frequency counts. The method is based on a deterministic streaming algorithm which efficiently computes approximate frequency counts over a stream of data while employing a small memory footprint. We show that this method easily scales to billion-word monolingual corpora using a conventional (8 GB RAM) desktop machine. Statistical machine translation experimental results corroborate that the resulting high-n approximate *small* language model is as effective as models obtained from other count pruning methods.

1 Introduction

In many NLP problems, we are faced with the challenge of dealing with large amounts of data. Many problems boil down to computing relative frequencies of certain items on this data. Items can be words, patterns, associations, n-grams, and others. Language modeling (Chen and Goodman, 1996), noun-clustering (Ravichandran et al., 2005), constructing syntactic rules for SMT (Galley et al., 2004), and finding analogies (Turney, 2008) are examples of some of the problems where we need to compute relative frequencies. We use language modeling as a canonical example of a large-scale task that requires relative frequency estimation.

Computing relative frequencies seems like an easy problem. However, as corpus sizes grow, it becomes a highly computational expensive task. Brants et al. (2007) used 1500 machines for a day to compute the relative frequencies of n-grams (summed over all orders from 1 to 5) from 1.8TB of web data. Their resulting model contained 300 million unique n-grams.

Cutoff	Size	BLEU	NIST	MET
Exact	367.6m	**28.73**	**7.691**	**56.32**
2	229.8m	28.23	7.613	56.03
3	143.6m	28.17	7.571	**56.53**
5	59.4m	28.33	7.636	56.03
10	18.3m	27.91	7.546	55.64
100	1.1m	28.03	7.607	55.91
200	0.5m	27.62	7.550	55.67

Table 1: Effect of count-based pruning on SMT performance using EAN corpus. Results are according to BLEU, NIST and METEOR (MET) metrics. Bold #s are not statistically significant worse than exact model.

It is not realistic using conventional computing resources to use all the 300 million n-grams for applications like speech recognition, spelling correction, information extraction, and statistical machine translation (SMT). Hence, one of the easiest way to reduce the size of this model is to use count-based pruning which discards all n-grams whose count is less than a pre-defined threshold. Although count-based pruning is quite simple, yet it is effective for machine translation. As we do not have a copy of the web, we will use a portion of gigaword i.e. EAN (see Section 4.1) to show the effect of count-based pruning on performance of SMT (see Section 5.1). Table 1 shows that using a cutoff of 100 produces a model of size 1.1 million n-grams with a Bleu score of 28.03. If we compare this with an exact model of size 367.6 million n-grams, we see an increase of 0.8 points in Bleu (95% statistical significance level

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 512–520,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

ϵ	Size	BLEU	NIST	MET
Exact	367.6m	**28.73**	**7.691**	**56.32**
1e-10	218.4m	**28.64**	**7.669**	**56.33**
5e-10	171.0m	**28.48**	**7.666**	**56.38**
1e-9	148.0m	**28.56**	7.646	**56.51**
5e-9	91.9m	28.27	7.623	56.16
1e-8	69.4m	28.15	7.609	56.19
5e-7	28.5m	28.08	7.595	55.91

Table 2: Effect of entropy-based pruning on SMT performance using EAN corpus. Results are as in Table 1

is ≈ 0.53 Bleu). However, we need 300 times bigger model to get such an increase. Unfortunately, it is not possible to integrate such a big model inside a decoder using normal computation resources.

A better way of reducing the size of n-grams is to use entropy pruning (Stolcke, 1998). Table 2 shows the results with entropy pruning with different settings of ϵ. We see that for three settings of ϵ equal to $1e$-10, $5e$-10 and $1e$-9, we get Bleu scores comparable to the exact model. However, the size of all these models is not at all small. The size of smallest model is 25% of the exact model. Even with this size it is still not feasible to integrate such a big model inside a decoder. If we take a model of size comparable to count cutoff of 100, i.e., with ϵ = 5e-7, we see both count-based pruning as well as entropy pruning performs the same.

There also have been prior work on maintaining approximate counts for higher-order language models (LMs) ((Talbot and Osborne, 2007a; Talbot and Osborne, 2007b; Talbot and Brants, 2008)) operates under the model that the goal is to store a compressed representation of a disk-resident table of counts and use this compressed representation to answer count queries approximately.

There are two difficulties with scaling all the above approaches as the order of the LM increases. Firstly, the computation time to build the database of counts increases rapidly. Secondly, the initial disk storage required to maintain these counts, prior to building the compressed representation is enormous.

The method we propose solves both of these problems. We do this by making use of the *streaming algorithm* paradigm (Muthukrishnan, 2005). Working under the assumption that multiple-GB models are infeasible, our goal is to instead of estimating a large model and then compressing it, we directly estimate a small model. We use a deterministic streaming algorithm (Manku and Motwani, 2002) that computes approximate frequency counts of frequently occurring n-grams. This scheme is considerably more accurate in getting the actual counts as compared to other schemes (Demaine et al., 2002; Karp et al., 2003) that find the set of frequent items without caring about the accuracy of counts.

We use these counts directly as features in an SMT system, and propose a direct way to integrate these features into an SMT decoder. Experiments show that directly storing approximate counts of frequent 5-grams compared to using count or entropy-based pruning counts gives equivalent SMT performance, while dramatically reducing the memory usage and getting rid of pre-computing a large model.

2 Background

2.1 n-gram Language Models

Language modeling is based on assigning probabilities to sentences. It can either compute the probability of an entire sentence or predict the probability of the next word in a sequence. Let w_1^m denote a sequence of words $(w_1, \ldots, w_m)$. The probability of estimating word w_m depends on previous n-1 words where n denotes the size of n-gram. This assumption that probability of predicting a current word depends on the previous words is called a Markov assumption, typically estimated by relative frequency:

$$P(w_m \mid w_{m-n+1}^{m-1}) = \frac{C(w_{m-n+1}^{m-1} w_m)}{C(w_{m-n+1}^{m-1})} \qquad (1)$$

Eq 1 estimates the n-gram probability by taking the ratio of observed frequency of a particular sequence and the observed frequency of the prefix. This is precisely the relative frequency estimate we seek.

2.2 Large-scale Language modeling

Using higher order LMs to improve the accuracy of SMT is not new. (Brants et al., 2007; Emami et al., 2007) built 5-gram LMs over web using distributed cluster of machines and queried them via network requests. Since the use of cluster of machines is not always practical, (Talbot and Osborne, 2007b; Talbot and Osborne, 2007a) showed a randomized data structure called Bloom filter, that can be used to construct space efficient language models

for SMT. (Talbot and Brants, 2008) presented randomized language model based on perfect hashing combined with entropy pruning to achieve further memory reductions. A problem mentioned in (Talbot and Brants, 2008) is that the algorithm that computes the compressed representation might need to retain the entire database in memory; in their paper, they design strategies to work around this problem. (Federico and Bertoldi, 2006) also used single machine and fewer bits to store the LM probability by using efficient prefix trees.

(Uszkoreit and Brants, 2008) used partially class-based LMs together with word-based LMs to improve SMT performance despite the large size of the word-based models used. (Schwenk and Koehn, 2008; Zhang et al., 2006) used higher language models at time of re-ranking rather than integrating directly into the decoder to avoid the overhead of keeping LMs in the main memory since disk lookups are simply too slow. Now using higher order LMs at time of re-ranking looks like a good option. However, the target n-best hypothesis list is not diverse enough. Hence if possible it is always better to integrate LMs directly into the decoder.

2.3 Streaming

Consider an algorithm that reads the input from a read-only *stream* from left to right, with no ability to go back to the input that it has already processed. This algorithm has working storage that it can use to store parts of the input or other intermediate computations. However, (and this is a critical constraint), this working storage space is significantly smaller than the input stream length. For typical algorithms, the storage size is of the order of $\log^k N$, where N is the input size and k is some constant.

Stream algorithms were first developed in the early 80s, but gained in popularity in the late 90s as researchers first realized the challenges of dealing with massive data sets. A good survey of the model and core challenges can be found in (Muthukrishnan, 2005). There has been considerable work on the problem of identifying high-frequency items (items with frequency above a threshold), and a detailed review of these methods is beyond the scope of this article. A new survey by (Cormode and Hadjieleftheriou, 2008) comprehensively reviews the literature.

3 Space-Efficient Approximate Frequency Estimation

Prior work on approximate frequency estimation for language models provide a "no-false-negative" guarantee, ensuring that counts for n-grams in the model are returned exactly, while working to make sure the false-positive rate remains small (Talbot and Osborne, 2007a). The notion of approximation we use is different: in our approach, it is the actual count values that will be approximated. We also exploit the fact that low-frequency n-grams, while constituting the vast majority of the set of unique n-grams, are usually smoothed away and are less likely to influence the language model significantly. Discarding low-frequency n-grams is particularly important in a stream setting, because it can be shown in general that any algorithm that generates approximate frequency counts for all n-grams requires space linear in the input stream (Alon et al., 1999).

We employ an algorithm for approximate frequency counting proposed by (Manku and Motwani, 2002) in the context of database management. Fix parameters $s \in (0, 1)$, and $\epsilon \in (0, 1), \epsilon \ll s$. Our goal is to approximately find all n-grams with frequency at least sN. For an input stream of n-grams of length N, the algorithm outputs a set of items (and frequencies) and guarantees the following:

- All items with frequencies exceeding sN are output (*no false negatives*).
- No item with frequency less than $(s - \epsilon)N$ is output (*few false positives*).
- All reported frequencies are less than the true frequencies by at most ϵN (*close-to-exact frequencies*).
- The space used by the algorithm is $O(\frac{1}{\epsilon} \log \epsilon N)$.

A simple example illustrates these properties. Let us fix $s = 0.01, \epsilon = 0.001$. Then the algorithm guarantees that all n-grams with frequency at least 1% will be returned, no element with frequency less than 0.9% will be returned, and all frequencies will be no more than 0.1% away from the true frequencies. The space used by the algorithm is $O(\log N)$, which can be compared to the much larger (close to N) space

needed to store the initial frequency counts. In addition, the algorithm runs in linear time by definition, requiring only one pass over the input. Note that there might be $\frac{1}{\epsilon}$ elements with frequency at least ϵN, and so the algorithm uses optimal space (up to a logarithmic factor).

3.1 The Algorithm

We present a high-level overview of the algorithm; for more details, the reader is referred to (Manku and Motwani, 2002). The algorithm proceeds by conceptually dividing the stream into *epochs*, each containing $1/\epsilon$ elements. Note that there are ϵN epochs. Each such epoch has an ID, starting from 1. The algorithm maintains a list of tuples[1] of the form (e, f, Δ), where e is an n-gram, f is its reported frequency, and Δ is the maximum error in the frequency estimation. While the algorithm reads n-grams associated with the current epoch, it does one of two things: if the new element e is contained in the list of tuples, it merely increments the frequency count f. If not, it creates a new tuple of the form $(e, 1, T-1)$, where T is the ID of the current epoch.

After each epoch, the algorithm "cleans house" by eliminating tuples whose maximum true frequency is small. Formally, if the epoch that just ended has ID T, then the algorithm deletes all tuples satisfying condition $f + \Delta \leq T$. Since $T \leq \epsilon N$, this ensures that no low-frequency tuples are retained. When all elements in the stream have been processed, the algorithm returns all tuples (e, f, Δ) where $f \geq (s-\epsilon)N$. In practice, however we do not care about s and return all tuples. At a high level, the reason the algorithm works is that if an element has high frequency, it shows up more than once each epoch, and so its frequency gets updated enough to stave off elimination.

4 Intrinsic Evaluation

We conduct a set of experiments with approximate n-gram counts (stream counts) produced by the stream algorithm. We define various metrics on which we evaluate the quality of stream counts compared with exact n-gram counts (true counts). To evaluate the quality of stream counts on these metrics, we carry out three experiments.

[1]We use hash tables to store tuples; however smarter data structures like suffix trees could also be used.

Corpus	Gzip-MB	M-wrds	Perplexity
EP	63	38	1122.69
afe	417	171	1829.57
apw	1213	540	1872.96
nyt	2104	914	1785.84
xie	320	132	1885.33

Table 3: Corpus Statistics and perplexity of LMs made with each of these corpuses on development set

4.1 Experimental Setup

The freely available English side of Europarl (EP) and Gigaword corpus (Graff, 2003) is used for computing n-gram counts. We only use EP along with two sections of the Gigaword corpus: Agence France Press English Service(afe) and The New York Times Newswire Service (nyt). The unigram language models built using these corpuses yield better perplexity scores on the development set (see Section 5.1) compared to The Xinhua News Agency English Service (xie) and Associated Press Worldstream English Service (apw) as shown in Table 3. The LMs are build using the SRILM language modelling toolkit (Stolcke, 2002) with modified Kneser-Ney discounting and interpolation. The evaluation of stream counts is done on EP+afe+nyt (EAN) corpus, consisting of 1.1 billion words.

4.2 Description of the metrics

To evaluate the quality of counts produced by our stream algorithm four different metrics are used. The accuracy metric measures the quality of top N stream counts by taking the fraction of top N stream counts that are contained in the top N true counts.

$$\text{Accuracy} = \frac{\text{Stream Counts} \cap \text{True Counts}}{\text{True Counts}}$$

Spearman's rank correlation coefficient or Spearman's rho(ρ) computes the difference between the ranks of each observation (i.e. n-gram) on two variables (that are top N stream and true counts). This measure captures how different the stream count ordering is from the true count ordering.

$$\rho = 1 - \frac{6 \sum d_i^2}{N(N^2-1)}$$

d_i is the difference between the ranks of corresponding elements X_i and Y_i; N is the number of elements found in both sets; X_i and Y_i in our case denote the stream and true counts.

Mean square error (MSE) quantifies the amount by which a predicted value differs from the true value. In our case, it estimates how different the stream counts are from the true counts.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\text{true}_i - \text{predicted}_i)^2$$

true and predicted denotes values of true and stream counts; N denotes the number of stream counts contained in true counts.

4.3 Varying ϵ experiments

In our first experiment, we use accuracy, ρ and MSE metrics for evaluation. Here, we compute 5-gram stream counts with different settings of ϵ on the EAN corpus. ϵ controls the number of stream counts produced by the algorithm. The results in Table 4 support the theory that decreasing the value of ϵ improves the quality of stream counts. Also, as expected, the algorithm produces more stream counts with smaller values of ϵ. The evaluation of stream counts obtained with ϵ = $50e$-8 and $20e$-8 reveal that the stream counts learned with this large value are more susceptible to errors.

If we look closely at the counts for ϵ = $50e$-8, we see that we get at least 30% of the stream counts from $245k$ true counts. This number is not significantly worse than the 36% of stream counts obtained from $4,018k$ true counts for the smallest value of $\epsilon = 5$e-8. However, if we look at the other two metrics, the ranking correlation ρ of stream counts compared with true counts on ϵ = $50e$-8 and $20e$-8 is low compared to other ϵ values. For the MSE, the error with stream counts on these ϵm values is again high compared to other values. As we decrease the value of ϵ we continually get better results: decreasing ϵ pushes the stream counts towards the true counts. However, using a smaller ϵ increases the memory usage. Looking at the evaluation, it is therefore advisable to use 5-gram stream counts produced with at most $\epsilon \leq 10e$-7 for the EAN corpus.

Since it is not possible to compute true 7-grams counts on EAN with available computing resources, we carry out a similar experiment for 7-grams on EP to verify the results for higher order n-grams [2]. The results in Table 5 tell a story similar to our results for 7-grams. The size of EP corpus is much smaller than EAN and so we see even better results on each of the metrics with decreasing the value of ϵ. The overall trend remains the same; here too, setting $\epsilon \leq 10e$-8 is the most effective strategy. The fact that these results are consistent across two datasets of different sizes and different n-gram sizes suggests that they will carry over to other tasks.

ϵ	5-**gram produced**	**Acc**	ρ	**MSE**
50e-8	245k	0.294	-3.6097	0.4954
20e-8	726k	0.326	-2.6517	0.1155
10e-8	1655k	0.352	-1.9960	0.0368
5e-8	4018k	0.359	-1.7835	0.0114

Table 4: Evaluating quality of 5-gram stream counts for different settings of ϵ on EAN corpus

ϵ	7-**gram produced**	**Acc**	ρ	**MSE**
50e-8	44k	0.509	0.3230	0.0341
20e-8	128k	0.596	0.5459	0.0063
10e-8	246k	0.689	0.7413	0.0018
5e-8	567k	0.810	0.8599	0.0004

Table 5: Evaluating quality of 7-gram stream counts for different settings of ϵ on EP corpus

4.4 Varying top K experiments

In the second experiment, we evaluate the quality of the top K (sorted by frequency) 5-gram stream counts. Here again, we use accuracy, ρ and MSE for evaluation. We fix the value of ϵ to $5e$-8 and compute 5-gram stream counts on the EAN corpus. We vary the value of K between $100k$ and $4,018k$ (i.e all the n-gram counts produced by the stream algorithm). The experimental results in Table 6 support the theory that stream count algorithm computes the exact count of most of the high frequency n-grams. Looking closer, we see that if we evaluate the algorithm on just the top $100k$ 5-grams (roughly 5% of all 5-grams produced), we see almost perfect results. Further, if we take the top $1,000k$ 5-grams (approximately 25% of all 5-grams) we again see excellent

[2]Similar evaluation scores are observed for 9-gram stream counts with different values of ϵ on EP corpus.

Top K	Accuracy	ρ	MSE
100k	0.994	0.9994	0.01266
500k	0.934	0.9795	0.0105
1000k	0.723	0.8847	0.0143
2000k	0.504	0.2868	0.0137
4018k	0.359	-1.7835	0.0114

Table 6: Evaluating top K sorted 5-gram stream counts for ϵ=5e-8 on EAN corpus

Top K	Accuracy	ρ	MSE
10k	0.996	0.9997	0.0015
20k	0.989	0.9986	0.0016
50k	0.950	0.9876	0.0016
100k	0.876	0.9493	0.0017
246k	0.689	0.7413	0.0018

Table 7: Evaluating top K sorted 7-gram stream counts for ϵ=10e-8 on EP corpus

performance on all metrics. The accuracy of the results decrease slightly, but the ρ and MSE metrics are not decreased that much in comparison. Performance starts to degrade as we get to $2,000k$ (over 50% of all 5-grams), a result that is not too surprising. However, even here we note that the MSE is low, suggesting that the frequencies of stream counts (found in top K true counts) are very close to the true counts. Thus, we conclude that the quality of the 5-gram stream counts produced for this value of ϵ is quite high (in relation to the true counts).

As before, we corroborate our results with higher order n-grams. We evaluate the quality of top K 7-gram stream counts on EP.[3] Since EP is a smaller corpus, we evaluate the stream counts produced by setting ϵ to $10e$-8. Here we vary the value of K between $10k$ and $246k$ (the total number produced by the stream algorithm). Results are shown in Table 7. As we saw earlier with 5-grams, the top $10k$ (i.e. approximately 5% of all 7-grams) are of very high quality. Results, and this remains true even when we increase K to $100k$. There is a drop in the accuracy and a slight drop in ρ, while the MSE remains the same. Taking all counts again shows a significant decrease in both accuracy and ρ scores, but this does not affect MSE scores significantly. Hence, the 7-gram stream counts i.e. $246k$ counts produced by ϵ = $10e$-8 are quite accurate when compared to the top $246k$ true counts.

4.5 Analysis of tradeoff between coverage and space

In our third experiment, we investigate whether a large LM can help MT performance. We evaluate the coverage of stream counts built on the EAN corpus on the test data for SMT experiments (see Section 5.1) with different values of ϵm. We compute the recall of each model against 3071 sentences of test data where recall is the fraction of number of n-grams of a dataset found in stream counts.

$$\text{Recall} = \frac{\text{Number of } n\text{-grams found in stream counts}}{\text{Number of } n\text{-grams in dataset}}$$

We build unigram, bigram, trigram, 5-gram and 7-gram with four different values of ϵ. Table 8 contains the `gzip` size of the count file and the recall of various different stream count n-grams. As expected, the recall with respect to true counts is maximum for unigrams, bigrams, trigrams and 5-grams. However the amount of space required to store all true counts in comparison to stream counts is extremely high: we need $4.8GB$ of compressed space to store all the true counts for 5-grams.

For unigram models, we see that the recall scores are good for all values of ϵ. If we compare the approximate stream counts produced by largest ϵ (which is worst) to all true counts, we see that the stream counts compressed size is 50 times smaller than the true counts size, and is only three points worse in recall. Similar trends hold for bigrams, although the loss in recall is higher. As with unigrams, the loss in recall is more than made up for by the memory savings (a factor of nearly 150). For trigrams, we see a 14 point loss in recall for the smallest ϵ, but a memory savings of 400 times. For 5-grams, the best recall value is .020 ($1.2k$ out of $60k$ 5-gram stream counts are found in the test set). However, compared with the true counts we only loss a recall of 0.05 ($4.3k$ out of $60k$) points but memory savings of 150 times. In extrinsic evaluations, we will show that integrating 5-gram stream counts with an SMT system performs slightly worse than the true counts, while dramatically reducing the memory usage.

[3] Similar evaluation scores are observed for different top K sorted 9-gram stream counts with ϵ=10e-8 on EP corpus.

N-gram	unigram		bigram		trigram		5-gram		7-gram	
ϵ	Gzip MB	Recall	Gzip MB	Recall	Gzip MB	Recall	Gzip MB	Recall	Gzip MB	Recall
50e-8	.352	.785	2.3	.459	3.3	.167	1.9	.006	.864	5.6e-5
20e-8	.568	.788	4.5	.494	7.6	.207	5.3	.011	2.7	1.3e-4
10e-8	.824	.791	7.6	.518	15	.237	13	.015	9.7	4.1e-4
5e-8	1.3	.794	13	.536	30	.267	31	.020	43	5.9e-4
all	17	.816	228	.596	1200	.406	4800	.072	NA	

Table 8: Gzipped space required to store n-gram counts on disk and their coverage on a test set with different ϵm

For 7-gram we can not compute the true n-gram counts due to limitations of available computational resources. The memory requirements with smallest value of ϵ are similar to those of 5-gram, but the recall values are quite small. For 7-grams, the best recall value is $5.9e$-4 which means that stream counts contains only 32 out of $54k$ 7-grams contained in test set. The small recall value for 7-grams suggests that these counts may not be that useful in SMT. We further substantiate our findings in our extrinsic evaluations. There we show that integrating 7-gram stream counts with an SMT system does not affect its overall performance significantly.

5 Extrinsic Evaluation

5.1 Experimental Setup

All the experiments conducted here make use of publicly available resources. Europarl (EP) corpus French-English section is used as parallel data. The publicly available Moses[4] decoder is used for training and decoding (Koehn and Hoang, 2007). The news corpus released for ACL SMT workshop in 2007 consisting of 1057 sentences[5] is used as the development set. Minimum error rate training (MERT) is used on this set to obtain feature weights to optimize translation quality. The final SMT system performance is evaluated on a uncased test set of 3071 sentences using the BLEU (Papineni et al., 2002), NIST (Doddington, 2002) and METEOR (Banerjee and Lavie, 2005) scores. The test set is the union of the 2007 news devtest and 2007 news test data from ACL SMT workshop 2007.[6]

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

[5]http://www.statmt.org/wmt07/

[6]We found that testing on Parliamentary test data was completely insensitive to large n-gram LMs, even when these LMs are exact. This suggests that for SMT performance, more data is better *only if* it comes from the right domain.

5.2 Integrating stream counts feature into decoder

Our method only computes high-frequency n-gram counts; it does not estimate conditional probabilities. We can either turn these counts into conditional probabilities (by using SRILM) or use the counts directly. We observed no significant difference in performance between these two approaches. However, using the counts directly consumes significantly less memory at run-time and is therefore preferable. Due to space constraints, SRILM results are omitted.

The only remaining open question is: *how should we turn the counts into a feature that can be used in an SMT system?* We considered several alternatives; the most successful was a simple weighted count of n-gram matches of varying size, appropriately backed-off. Specifically, consider an n-gram model. For every sequence of words $w_i, \ldots, w_{i+N-1}$, we obtain a feature score computed recursively according to Eq (2).

$$\begin{aligned} f(w_i) &= \log\left(\frac{C(w_i)}{Z}\right) \\ f(w_i, \ldots, w_{i+k}) &= \log\left(\frac{C(w_i, \ldots, w_{i+k})}{Z}\right) \\ &+ \frac{1}{2} f(w_{i+1}, \ldots, w_{i+k}) \end{aligned} \quad (2)$$

Here, $\frac{1}{2}$ is the backoff factor and Z is the largest count in the count set (the presence of Z is simply to ensure that these values remain manageable). In order to efficiently compute these features, we store the counts in a suffix-tree. The computation proceeds by first considering w_{i+N-1} alone and then "expanding" to consider the bigram, then trigram and so on. The advantage to this order of computation is that the recursive calls can cease whenever a

n-gram(ϵ)	BLEU	NIST	MET	Mem GB
3 EP(exact)	25.57	7.300	54.48	2.7
5 EP(exact)	25.79	7.286	54.44	2.9
3 EAN(exact)	27.04	7.428	55.07	4.6
5 EAN(exact)	**28.73**	**7.691**	**56.32**	20.5
4(10e-8)	27.36	7.506	**56.19**	2.7
4(5e-8)	27.40	7.507	55.90	2.8
5(10e-8)	27.97	7.605	55.52	2.8
5(5e-8)	27.98	**7.611**	56.07	2.8
7(10e-8)	27.97	7.590	55.88	2.9
7(5e-8)	27.88	7.577	56.01	2.9
9(10e-8)	**28.18**	**7.611**	55.95	2.9
9(5e-8)	27.98	7.608	56.08	2.9

Table 9: Evaluating SMT with different LMs on EAN. Results are according to BLEU, NIST and MET metrics. Bold #s are not statistically significant worse than exact.

zero count is reached. (Extending Moses to include this required only about 100 lines of code.)

5.3 Results

Table 9 summarizes SMT results. We have 4 baseline LMs that are conventional LMs smoothed using modified Kneser-Ney smoothing. The first two trigram and 5-gram LMs are built on EP corpus and the other two are built on EAN corpus. Table 9 show that there is not much significant difference in SMT results of 5-gram and trigram LM on EP. As expected, the trigram built on the large corpus EAN gets an improvement of 1.5 Bleu Score. However, unlike the EP corpus, building a 5-gram LM on EAN (huge corpus) gets an improvement of 3.2 Bleu Score. (The 95% statistical significance boundary is about $\pm$ 0.53 Bleu on the test data, 0.077 Nist and 0.16 Meteor according to bootstrap resampling) We see similar gains in Nist and Meteor metrics as shown in Table 9.

We use stream counts computed with two values of ϵ, 5e-8 and 10e-8 on EAN corpus. We use all the stream counts produced by the algorithm. 4, 5, 7 and 9 order n-gram stream counts are computed with these settings of ϵ. These counts are used along with a trigram LM built on EP to improve SMT performance. The memory usage (Mem) shown in Table 9 is the full memory size required to run on the test data (including phrase tables).

Adding 4-gram and 5-gram stream counts as feature helps the most. The performance gain by using 5-gram stream counts is slightly worse than compared to true 5-gram LM on EAN. However, using 5-gram stream counts directly is more memory efficient. Also, the gains for stream counts are exactly the same as we saw for same sized count-based and entropy-based pruning counts in Table 1 and 2 respectively. Moreover, unlike the pruning methods, our algorithm directly computes a small model, as opposed to compressing a pre-computed large model.

Adding 7-gram and 9-gram does not help significantly, a fact anticipated by the low recall of 7-gram-based counts that we saw in Section 4.5. The results with two different settings of ϵ are largely the same. This validates our intrinsic evaluation results in Section 4.3 that stream counts learned using $\epsilon \leq$ 10e-8 are of good quality, and that the quality of the stream counts is high.

6 Conclusion

We have proposed an efficient, low-memory method to construct high-order approximate n-gram LMs. Our method easily scales to billion-word monolingual corpora on conventional (8GB) desktop machines. We have demonstrated that approximate n-gram features could be used as a direct replacement for conventional higher order LMs in SMT with significant reductions in memory usage. In future, we will be looking into building streaming skip n-grams, and other variants (like cluster n-grams).

In NLP community, it has been shown that having more data results in better performance (Ravichandran et al., 2005; Brants et al., 2007; Turney, 2008). At web scale, we have terabytes of data and that can capture broader knowledge. Streaming algorithm paradigm provides a memory and space-efficient platform to deal with terabytes of data. We hope that other NLP applications (where we need to compute relative frequencies) like noun-clustering, constructing syntactic rules for SMT, finding analogies, and others can also benefit from streaming methods. We also believe that stream counts can be applied to other problems involving higher order LMs such as speech recognition, information extraction, spelling correction and text generation.

References

Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1).

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

S. Chen and J. Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of 34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, CA, June.

Graham Cormode and Marios Hadjieleftheriou. 2008. Finding frequent items in data streams. In *VLDB*.

E.D. Demaine, A. Lopez-Ortiz, and J.I. Munro. 2002. Frequency estimation of internet packet streams with limited space.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*.

Ahmad Emami, Kishore Papineni, and Jeffrey Sorensen. 2007. Large-scale distributed language modeling. In *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages 37–40.

Marcello Federico and Nicola Bertoldi. 2006. How many bits are needed to store probabilities for phrase-based translation? In *Proceedings on the Workshop on Statistical Machine Translation at ACL06*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL-04*.

D. Graff. 2003. English Gigaword. Linguistic Data Consortium, Philadelphia, PA, January.

Richard M. Karp, Christos H. Papadimitriou, and Scott Shenker. 2003. A simple algorithm for finding frequent elements in streams and bags.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876.

G. S. Manku and R. Motwani. 2002. Approximate frequency counts over data streams. In *Proceedings of the 28th International Conference on Very Large Data Bases*.

S. Muthukrishnan. 2005. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2).

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*.

Holger Schwenk and Philipp Koehn. 2008. Large and diverse language models for statistical machine translation. In *Proceedings of The Third International Joint Conference on Natural Language Processing (IJCNP)*.

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

A. Stolcke. 2002. SRILM – An Extensible Language Modeling Toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 901–904, Denver, CO, September.

David Talbot and Thorsten Brants. 2008. Randomized language models via perfect hash functions. In *Proceedings of ACL-08: HLT*.

David Talbot and Miles Osborne. 2007a. Randomised language modelling for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*.

David Talbot and Miles Osborne. 2007b. Smoothed Bloom filter language models: Tera-scale LMs on the cheap. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EM NLP-CoNLL)*.

Peter D. Turney. 2008. A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of COLING 2008*.

Jakob Uszkoreit and Thorsten Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of ACL-08: HLT*.

Ying Zhang, Almut Silja Hildebrand, and Stephan Vogel. 2006. Distributed language modeling for n-best list re-ranking. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*.

The Effect of Corpus Size on Case Frame Acquisition for Discourse Analysis

Ryohei Sasano
Graduate School of Informatics, Kyoto University
sasano@i.kyoto-u.ac.jp

Daisuke Kawahara
National Institute of Information and Communications Technology
dk@nict.go.jp

Sadao Kurohashi
Graduate School of Informatics, Kyoto University
kuro@i.kyoto-u.ac.jp

Abstract

This paper reports the effect of corpus size on case frame acquisition for discourse analysis in Japanese. For this study, we collected a Japanese corpus consisting of up to 100 billion words, and constructed case frames from corpora of six different sizes. Then, we applied these case frames to syntactic and case structure analysis, and zero anaphora resolution. We obtained better results by using case frames constructed from larger corpora; the performance was not saturated even with a corpus size of 100 billion words.

1 Introduction

Very large corpora obtained from the Web have been successfully utilized for many natural language processing (NLP) applications, such as prepositional phrase (PP) attachment, other-anaphora resolution, spelling correction, confusable word set disambiguation and machine translation (Volk, 2001; Modjeska et al., 2003; Lapata and Keller, 2005; Atterer and Schütze, 2006; Brants et al., 2007).

Most of the previous work utilized only the surface information of the corpora, such as n-grams, co-occurrence counts, and simple surface syntax. This may be because these studies did not require structured knowledge, and for such studies, the size of currently available corpora is considered to have been almost enough. For instance, while Brants et al. (2007) reported that translation quality continued to improve with increasing corpus size for training language models at even size of 2 trillion tokens, the increase became small at the corpus size of larger than 30 billion tokens.

However, for more complex NLP tasks, such as case structure analysis and zero anaphora resolution, it is necessary to obtain more structured knowledge, such as semantic case frames, which describe the cases each predicate has and the types of nouns that can fill a case slot. Note that case frames offer not only the knowledge of the relationships between a predicate and its particular case slot, but also the knowledge of the relationships among a predicate and its multiple case slots. To obtain such knowledge, very large corpora seem to be necessary; however it is still unknown how much corpora would be required to obtain good coverage.

For examples, Kawahara and Kurohashi proposed a method for constructing wide-coverage case frames from large corpora (Kawahara and Kurohashi, 2006b), and a model for syntactic and case structure analysis of Japanese that based upon case frames (Kawahara and Kurohashi, 2006a). However, they did not demonstrate whether the coverage of case frames was wide enough for these tasks and how dependent the performance of the model was on the corpus size for case frame construction.

This paper aims to address these questions. We collect a very large Japanese corpus consisting of about 100 billion words, or 1.6 billion unique sentences from the Web. Subsets of the corpus are randomly selected to obtain corpora of different sizes ranging from 1.6 million to 1.6 billion sentences. We construct case frames from each corpus and apply them to syntactic and case structure analysis, and zero anaphora resolution, in order to investigate the

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 521–529,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

relationships between the corpus size and the performance of these analyses.

2 Related Work

Many NLP tasks have successfully utilized very large corpora, most of which were acquired from the Web (Kilgarriff and Grefenstette, 2003). Volk (2001) proposed a method for resolving PP attachment ambiguities based upon Web data. Modjeska et al. (2003) used the Web for resolving nominal anaphora. Lapata and Keller (2005) investigated the performance of web-based models for a wide range of NLP tasks, such as MT candidate selection, article generation, and countability detection. Nakov and Hearst (2008) solved relational similarity problems using the Web as a corpus.

With respect to the effect of corpus size on NLP tasks, Banko and Brill (2001a) showed that for content sensitive spelling correction, increasing the training data size improved the accuracy. Atterer and Schütze (2006) investigated the effect of corpus size in combining supervised and unsupervised learning for two types of attachment decision; they found that the combined system only improved the performance of the parser for small training sets. Brants et al. (2007) varied the amount of language model training data from 13 million to 2 trillion tokens and applied these models to machine translation systems. They reported that translation quality continued to improve with increasing corpus size for training language models at even size of 2 trillion tokens. Suzuki and Isozaki (2008) provided evidence that the use of more unlabeled data in semi-supervised learning could improve the performance of NLP tasks, such as POS tagging, syntactic chunking, and named entities recognition.

There are several methods to extract useful information from very large corpora. Search engines, such as Google and Altavista, are often used to obtain Web counts (e.g. (Nakov and Hearst, 2005; Gledson and Keane, 2008)). However, search engines are not designed for NLP research and the reported hit counts are subject to uncontrolled variations and approximations. Therefore, several researchers have collected corpora from the Web by themselves. For English, Banko and Brill (2001b) collected a corpus with 1 billion words from variety of English texts. Liu and Curran (2006) created a Web corpus for English that contained 10 billion words and showed that for content-sensitive spelling correction the Web corpus results were better than using a search engine. Halacsy et al. (2004) created a corpus with 1 billion words for Hungarian from the Web by downloading 18 million pages. Others utilize publicly available corpus such as the North American News Corpus (NANC) and the Gigaword Corpus (Graff, 2003). For instance, McClosky et al. (2006) proposed a simple method of self-training a two phase parser-reranker system using NANC.

As for Japanese, Kawahara and Kurohashi (2006b) collected 23 million pages and created a corpus with approximately 20 billion words. Google released Japanese n-gram constructed from 20 billion Japanese sentences (Kudo and Kazawa, 2007). Several news wires are publicly available consisting of tens of million sentences. Kotonoha project is now constructing a balanced corpus of the present-day written Japanese consisting of 50 million words (Maekawa, 2006).

3 Construction of Case Frames

Case frames describe the cases each predicate has and what nouns can fill the case slots. In this study, case frames we construct case frames from raw corpora by using the method described in (Kawahara and Kurohashi, 2006b). This section illustrates the methodology for constructing case frames.

3.1 Basic Method

After parsing a large corpus by a Japanese parser KNP[1], we construct case frames from modifier-head examples in the resulting parses. The problems for case frame construction are syntactic and semantic ambiguities. In other words, the resulting parses inevitably contain errors and predicate senses are intrinsically ambiguous. To cope with these problems, we construct case frames from reliable modifier-head examples.

First, we extract modifier-head examples that had no syntactic ambiguity, and assemble them by coupling a predicate and its closest case component. That is, we assemble the examples not by predicates, such as *tsumu* (load/accumulate), but by cou-

[1] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/knp-e.html

Table 1: Examples of Constructed Case Frames.

	Case slot	Examples	Generalized examples with rate
tsumu (1) (load)	*ga* (nominative)	he, driver, friend, ···	[CT:PERSON]:0.45, [NE:PERSON]:0.08, ···
	wo (accusative)	baggage, luggage, hay, ···	[CT:ARTIFACT]:0.31, ···
	ni (dative)	car, truck, vessel, seat, ···	[CT:VEHICLE]:0.32, ···
tsumu (2) (accumulate)	*ga* (nominative)	player, children, party, ···	[CT:PERSON]:0.40, [NE:PERSON]:0.12, ···
	wo (accusative)	experience, knowledge, ···	[CT:ABSTRACT]:0.47, ···
⋮	⋮	⋮	
hanbai (1) (sell)	*ga* (nominative)	company, Microsoft, firm, ···	[NE:ORGANIZATION]:0.16, [CT:ORGANIZATION]:0.13, ···
	wo (accusative)	goods, product, ticket, ···	[CT:ARTIFACT]:0.40, [CT:FOOD]:0.07, ···
	ni (dative)	customer, company, user, ···	[CT:PERSON]:0.28, ···
	de (locative)	shop, bookstore, site ···	[CT:FACILITY]:0.40, [CT:LOCATION]:0.39, ···
⋮	⋮	⋮	

ples, such as *nimotsu-wo tsumu* (load baggage) and *keiken-wo tsumu* (accumulate experience). Such couples are considered to play an important role for constituting sentence meanings. We call the assembled examples as basic case frames. In order to remove inappropriate examples, we introduce a threshold α and use only examples that appeared no less than α times in the corpora.

Then, we cluster the basic case frames to merge similar case frames. For example, since *nimotsu-wo tsumu* (load baggage) and *busshi-wo tsumu* (load supplies) are similar, they are merged. The similarity is measured by using a Japanese thesaurus (The National Language Institute for Japanese Language, 2004). Table 1 shows examples of constructed case frames.

3.2 Generalization of Examples

When we use hand-crafted case frames, the data sparseness problem is serious; by using case frames automatically constructed from a large corpus, it was alleviated to some extent but not eliminated. For instance, there are thousands of named entities (NEs) that cannot be covered intrinsically. To deal with this problem, we generalize the examples of the case slots. Kawahara and Kurohashi also generalized examples but only for a few types. In this study, we generalize case slot examples based upon common noun categories and NE classes.

First, we generalize the examples based upon the categories that tagged by the Japanese morphological analyzer JUMAN[2]. In JUMAN, about 20 categories are defined and tagged to common nouns. For example, *ringo* (apple), *inu* (dog) and *byoin* (hospital) are tagged as FOOD, ANIMAL and FACILITY, respectively. For each category, we calculate the ratio of the categorized example among all case slot examples, and add it to the case slot (e.g. [CT:FOOD]:0.07).

[2]http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman-e.html

Table 2: Definition of NE in IREX.

NE class	Examples
ORGANIZATION	NHK Symphony Orchestra
PERSON	Kawasaki Kenjiro
LOCATION	Rome, Sinuiju
ARTIFACT	Nobel Prize
DATE	July 17, April this year
TIME	twelve o'clock noon
MONEY	sixty thousand dollars
PERCENT	20%, thirty percents

We also generalize the examples based upon NE classes. We use a common standard NE definition for Japanese provided by the IREX (1999). We first recognize NEs in the source corpus by using an NE recognizer (Sasano and Kurohashi, 2008); and then construct case frames from the NE-recognized corpus. Similar to the categories, for each NE class, we calculate the NE ratio among all the case slot examples, and add it to the case slot (e.g. [NE:PERSON]:0.12). The generalized examples are also included in Table 1.

4 Discourse Analysis with Case Frames

In order to investigate the effect of corpus size on complex NLP tasks, we apply the constructed cases frames to an integrated probabilistic model for Japanese syntactic and case structure analysis (Kawahara and Kurohashi, 2006a) and a probabilistic model for Japanese zero anaphora resolution (Sasano et al., 2008). In this section, we briefly describe these models.

4.1 Model for Syntactic and Case Structure Analysis

Kawahara and Kurohashi (2006a) proposed an integrated probabilistic model for Japanese syntactic and case structure analysis based upon case frames. Case structure analysis recognizes predicate argument structures. Their model 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 $P(T, L|S)$:

$$(T_{best}, L_{best}) = \underset{(T,L)}{\operatorname{argmax}} P(T, L|S) = \underset{(T,L)}{\operatorname{argmax}} P(T, L, S) \quad (1)$$

The last equation is derived because $P(S)$ is constant. $P(T, L, S)$ is defined as the product of a probability for generating a clause C_i as follows:

$$P(T, L, S) = \prod_{i=1..n} P(C_i|b_{h_i}) \quad (2)$$

where n is the number of clauses in S, and b_{h_i} is C_i's modifying *bunsetsu*[3]. $P(C_i|b_{h_i})$ is approximately decomposed into the product of several generative probabilities such as $P(A(s_j) = 1|CF_l, s_j)$ and $P(n_j|CF_l,\ s_j, A(s_j) = 1)$, where the function $A(s_j)$ returns 1 if a case slot s_j is filled with an input case component; otherwise 0. $P(A(s_j)=1|CF_l, s_j)$ denotes the probability that the case slot s_j is filled with an input case component, and is estimated from resultant case structure analysis of a large raw corpus. $P(n_j|CF_l,\ s_j, A(s_j) = 1)$ denotes the probability of generating a content part n_j from a filled case slot s_j in a case frame CF_l, and is calculated by using case frames. For details see (Kawahara and Kurohashi, 2006a).

4.2 Model for Zero Anaphora Resolution

Anaphora resolution is one of the most important techniques for discourse analysis. In English, overt pronouns such as *she* and definite noun phrases such as *the company* are anaphors that refer to preceding entities (antecedents). On the other hand, in Japanese, anaphors are often omitted; these omissions are called *zero pronouns*. Zero anaphora resolution is the integrated task of zero pronoun detection and zero pronoun resolution.

We proposed a probabilistic model for Japanese zero anaphora resolution based upon case frames (Sasano et al., 2008). This model first resolves coreference and identifies discourse entities; then gives a probability to each possible case frame CF and case assignment CA when target predicate v, input case components ICC and existing discourse entities ENT are given, and outputs the case frame and case assignment that have the highest probability. That is to say, this model selects the case frame CF_{best} and the case assignment CA_{best} that maximize the probability $P(CF, CA|v, ICC, ENT)$:

$$(CF_{best}, CA_{best}) = \underset{(CF,CA)}{\operatorname{argmax}} P(CF, CA|v, ICC, ENT) \quad (3)$$

$P(CF, CA|v, ICC, ENT)$ is approximately decomposed into the product of several probabilities. Case frames are used for calculating $P(n_j|CF_l,\ s_j, A(s_j) = 1)$, the probability of generating a content part n_j from a case slot s_j in a case frame CF_l, and $P(n_j|CF_l, s_j, A'(s_j)=1)$, the probability of generating a content part n_j of a zero pronoun, where the function $A'(s_j)$ returns 1 if a case slot s_j is filled with an antecedent of a zero pronoun; otherwise 0.

$P(n_j|CF_l, s_j, A'(s_j)=1)$ is similar to $P(n_j|CF_l,\ s_j, A(s_j)=1)$ and estimated from the frequencies of case slot examples in case frames. However, while $A'(s_j)=1$ means s_j is not filled with an overt argument but filled with an antecedent of zero pronoun, case frames are constructed from overt predicate argument pairs. Therefore, the content part n_j is often not included in the case slot examples. To cope with this problem, this model also utilizes generalized examples to estimate $P(n_j|CF_l,\ s_j, A(s_j) = 1)$. For details see (Sasano et al., 2008).

5 Experiments

5.1 Construction of Case Frames

In order to investigate the effect of corpus size, we constructed case frames from corpora of different sizes. We first collected Japanese sentences

[3] 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.

Table 4: Statistics of the Constructed Case Frames.

Corpus size (sentences)		1.6M	6.3M	25M	100M	400M	1.6G
# of predicate		2460	6134	13532	27226	42739	65679
(type)	verb	2039	4895	10183	19191	28523	41732
	adjective	154	326	617	1120	1641	2318
	noun with copula	267	913	2732	6915	12575	21629
average # of case frames for a predicate		15.9	12.2	13.3	16.1	20.5	25.3
average # of case slots for a case frame		2.95	3.44	3.88	4.21	4.69	5.08
average # of examples for a case slot		4.89	10.2	19.5	34.0	67.2	137.6
average # of unique examples for a case slot		1.19	1.85	3.06	4.42	6.81	9.64
average # of generalized examples for a case slot		0.14	0.24	0.37	0.49	0.67	0.84
File size(byte)		8.9M	20M	56M	147M	369M	928M

Table 3: Corpus Sizes and Thresholds.

Corpus size for case frame construction (sentences)	1.6M	6.3M	25M	100M	400M	1.6G
Threshold α introduced in Sec. 3.1	2	3	4	5	7	10
Corpus size to estimate generative probability (sentences)	1.6M	3.2M	6.3M	13M	25M	50M

from the Web using the method proposed by Kawahara and Kurohashi (2006b). We acquired approximately 6 billion Japanese sentences consisting of approximately 100 billion words from 100 million Japanese web pages. After discarding duplicate sentences, which may have been extracted from mirror sites, we acquired a corpus comprising of 1.6 billion (1.6G) unique Japanese sentences consisting of approximately 25 billion words. The average number of characters and words in each sentence was 28.3, 15.6, respectively. Then we randomly selected subsets of the corpus for five different sizes; 1.6M, 6.3M, 25M, 100M, and 400M sentences to obtain corpora of different sizes.

We constructed case frames from each corpus. We employed JUMAN and KNP to parse each corpus. We changed the threshold α introduced in Section 3.1 depending upon the size of the corpus as shown in Table 3. Completing the case frame construction took about two weeks using 600 CPUs. Table 4 shows the statistics for the constructed case frames. The number of predicates, the average number of examples and unique examples for a case slot, and whole file size were confirmed to be heavily dependent upon the corpus size. However, the average number of case frames for a predicate and case slots for a case frame did not.

5.2 Coverage of Constructed Case Frames

5.2.1 Setting

In order to investigate the coverage of the resultant case frames, we used a syntactic relation, case structure, and anaphoric relation annotated corpus consisting of 186 web documents (979 sentences). This corpus was manually annotated using the same criteria as Kawahara et al. (2004). There were 2,390 annotated relationships between predicates and their direct (not omitted) case components and 837 zero anaphoric relations in the corpus.

We used two evaluation metrics depending upon whether the target case component was omitted or not. For the overt case component of a predicate, we judged the target component was covered by case frames if the target component itself was included in the examples for one of the corresponding case slots of the case frame. For the omitted case component, we checked not only the target component itself but also all mentions that refer to the same entity as the target component.

5.2.2 Coverage of Case Frames

Figure 1 shows the coverage of case frames for the overt argument, which would have tight relations with case structure analysis. The lower line shows the coverage without considering generalized examples, the middle line shows the coverage considering generalized NE examples, and the upper line shows the coverage considering all generalized examples.

Figure 2 shows the coverage of case frames for the omitted argument, which would have tight relations with zero anaphora resolution. The upper line shows the coverage considering all generalized examples, which is considered to be the upper bound of performance for the zero anaphora resolution sys-

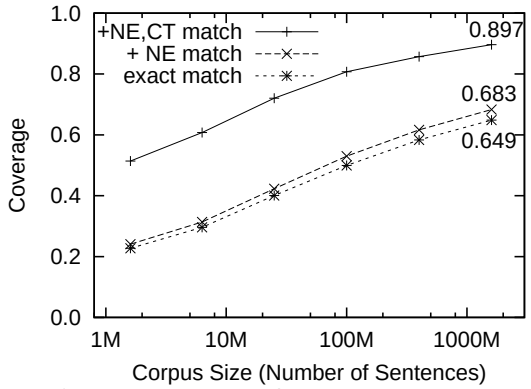


Figure 1: Coverage of CF (overt argument).

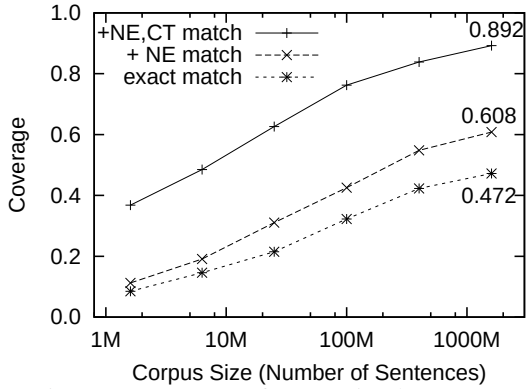


Figure 2: Coverage of CF (omitted argument).

tem described in Section 4.2. Comparing with Figure 1, we found two characteristics. First, the lower and middle lines of Figure 2 were located lower than the corresponding lines in Figure 1. This would reflect that some frequently omitted case components are not described in the case frames because the case frames were constructed from only overt predicate argument pairs. Secondly, the effect of generalized NE examples was more evident for the omitted argument reflecting the important role of NEs in zero anaphora resolution.

Both figures show that the coverage was improved by using larger corpora and there was no saturation even when the largest corpus of 1.6 billion sentences was used. When the largest corpus and all generalized examples were used, the case frames achieved a coverage of almost 90% for both the overt and omitted argument.

Figure 3 shows the coverage of case frames for each predicate type, which was calculated for both overt and omitted argument considering all generalized examples. The case frames for verbs achieved a coverage of 93.0%. There were 189 predicate-argument pairs that were not included case frames;

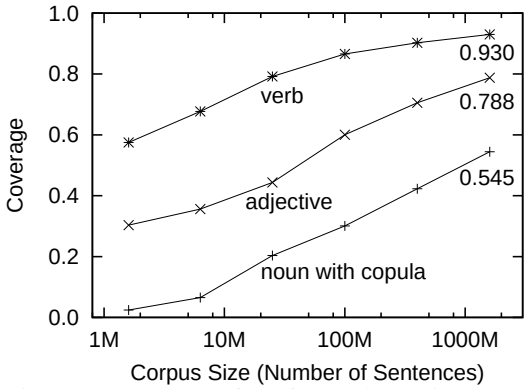


Figure 3: Coverage of CF for Each Predicate Type.

11 pairs of them were due to lack of the case frame of target predicate itself, and the others were due to lack of the corresponding example. For adjective, the coverage was 78.8%. The main cause of the lower coverage would be that the predicate argument relations concerning adjectives that were used in restrictive manner, such as "*oishii sushi*" (delicious sushi), were not used for case frame construction, although such relations were also the target of the coverage evaluation. For noun with copula, the coverage was only 54.5%. However, most predicate argument relations concerning nouns with copula were easily recognized from syntactic preference, and thus the low coverage would not quite affect the performance of discourse analysis.

5.3 Syntactic and Case Structure Analysis

5.3.1 Accuracy of Syntactic Analysis

We investigated the effect of corpus size for syntactic analysis described in Section 4.1. We used hand-annotated 759 web sentences, which was used by Kawahara and Kurohashi (2007). We evaluated the resultant syntactic structures with regard to dependency accuracy, the proportion of correct dependencies out of all dependencies[4].

Figure 4 shows the accuracy of syntactic structures. We conducted these experiments with case frames constructed from corpora of different sizes. We also changed the corpus size to estimate generative probability of a case slot in Section 4.1 depending upon the size of the corpus for case frame construction as shown in Table 3. Figure 4 also in-

[4]Note that Kawahara and Kurohashi (2007) exclude the dependency between the last two *bunsetsu*, since Japanese is head-final and thus the second last *bunsetsu* unambiguously depends on the last *bunsetsu*.

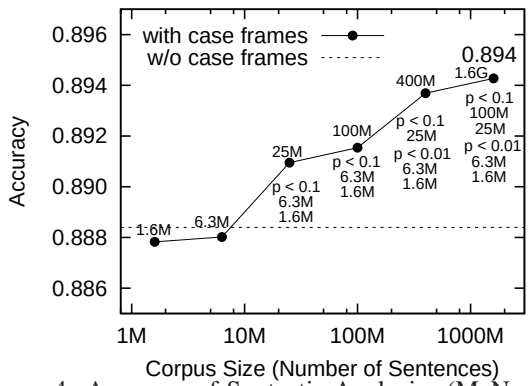


Figure 4: Accuracy of Syntactic Analysis. (McNemar's test results are also shown under each data point.)

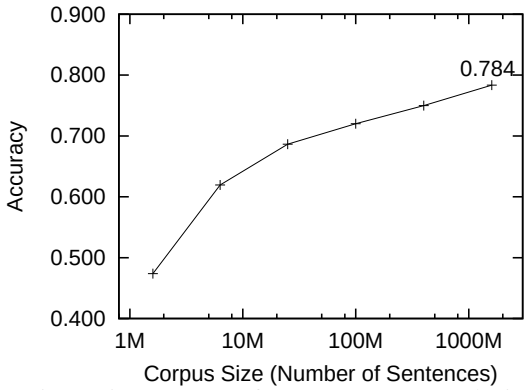


Figure 5: Accuracy of Case Structure Analysis.

cludes McNemar's test results. For instance, the difference between the corpus size of 1.6G and 100M sentences is significant at the 90% level ($p = 0.1$), but not significant at the 99% level ($p = 0.01$).

In Figure 4, 'w/o case frames' shows the accuracy of the rule-based syntactic parser KNP that does not use case frames. Since the model described in Section 4.1 assumes the existence of reasonable case frames, when we used case frames constructed from very small corpus, such as 1.6M and 6.3M sentences, the accuracy was lower than that of the rule-based syntactic parser. Moreover, when we tested the model described in Section 4.1 without any case frames, the accuracy was 0.885.

We confirmed that better performance was obtained by using case frames constructed from larger corpora, and the accuracy of 0.894[5] was achieved by using the case frames constructed from 1.6G sentences. However the effect of the corpus size was limited. This is because there are various causes of dependency error and the case frame sparseness problem is not serious for syntactic analysis.

We considered that generalized examples can benefit for the accuracy of syntactic analysis, and tried several models that utilize these examples. However, we cannot confirm any improvement.

5.3.2 Accuracy of Case Structure Analysis

We conducted case structure analysis on 215 web sentences in order to investigate the effect of corpus size for case structure analysis. The case markers of topic marking phrases and clausal modifiers were evaluated by comparing them with the gold standard in the corpus. Figure 5 shows the experimental results. We confirmed that the accuracy of case structure analysis strongly depends on corpus size for case frame construction.

[5]It corresponds to 0.877 in Kawahara and Kurohashi's (2007) evaluation metrics.

Table 5: Corpus Sizes for Case Frame Construction and Time for Syntactic and Case Structure Analysis.

Corpus size	1.6M	6.3M	25M	100M	400M	1.6G
Time (sec.)	850	1244	1833	2696	3783	5553

5.3.3 Analysis Speed

Table 5 shows the time for analyzing syntactic and case structure of 759 web sentences. Although the time for analysis became longer by using case frames constructed from a larger corpus, the growth rate was smaller than the growth rate of the size for case frames described in Table 4.

Since there is enough increase in accuracy of case structure analysis, we can say that case frames constructed larger corpora are desirable for case structure analysis.

5.4 Zero Anaphora Resolution

5.4.1 Accuracy of Zero Anaphora Resolution

We used an anaphoric relation annotated corpus consisting of 186 web documents (979 sentences) to evaluate zero anaphora resolution. We used first 51 documents for test and used the other 135 documents for calculating several probabilities. In the 51 test documents, 233 zero anaphora relations were annotated between one of the mentions of the antecedent and corresponding predicate that had zero pronoun.

In order to concentrate on evaluation for zero anaphora resolution, we used the correct mor-

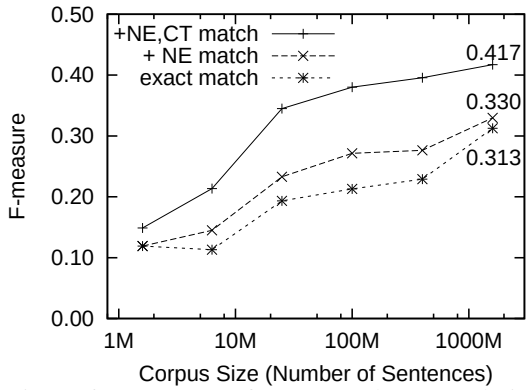


Figure 6: F-measure of Zero Anaphora Resolution.

phemes, named entities, syntactic structures and coreference relations that were manually annotated. Since correct coreference relations were given, the number of created entities was the same between the gold standard and the system output because zero anaphora resolution did not create new entities.

The experimental results are shown in Figure 6, in which F-measure was calculated by:

$$R = \frac{\text{\# of correctly recognized zero anaphora}}{\text{\# of zero anaphora annotated in corpus}},$$
$$P = \frac{\text{\# of correctly recognized zero anaphora}}{\text{\# of system outputted zero anaphora}},$$
$$F = \frac{2}{1/R + 1/P}.$$

The upper line shows the performance using all generalized examples, the middle line shows the performance using only generalized NEs, and the lower line shows the performance without using any generalized examples. While generalized categories much improved the F-measure, generalized NEs contributed little. This tendency is similar to that of coverage of case frames for omitted argument shown in Figure 2. Unlike syntactic and case structure analysis, the performance for the zero anaphora resolution is quite low when using case frames constructed from small corpora, and we can say case frames constructed from larger corpora are essential for zero anaphora resolution.

5.4.2 Analysis Speed

Table 6 shows the time for resolving zero anaphora in 51 web documents consisting of 278 sentences. The time for analysis became longer by using case frames constructed from larger corpora, which tendency is similar to the growth of the time for analyzing syntactic and case structure.

Table 6: Corpus Sizes for Case Frame Construction and Time for Zero Anaphora Resolution.

Corpus size	1.6M	6.3M	25M	100M	400M	1.6G
Time (sec.)	538	545	835	1040	1646	2219

5.5 Discussion

Experimental results of both case structure analysis and zero anaphora resolution show the effectiveness of a larger corpus in case frame acquisition for Japanese discourse analysis. Up to the corpus size of 1.6 billion sentences, or 100 billion words, these experimental results still show a steady increase in performance. That is, we can say that the corpus size of 1.6 billion sentences is not enough to obtain case frames of sufficient coverage.

These results suggest that increasing corpus size is more essential for acquiring structured knowledge than for acquiring unstructured statistics of a corpus, such as n-grams, and co-occurrence counts; and for complex NLP tasks such as case structure analysis and zero anaphora resolution, the currently available corpus size is not sufficient.

Therefore, to construct more wide-coverage case frames by using a larger corpus and reveal how much corpora would be required to obtain sufficient coverage is considered as future work.

6 Conclusion

This paper has reported the effect of corpus size on case frame acquisition for syntactic and case structure analysis, and zero anaphora resolution in Japanese. We constructed case frames from corpora of six different sizes ranging from 1.6 million to 1.6 billion sentences; and then applied these case frames to Japanese syntactic and case structure analysis, and zero anaphora resolution. Experimental results showed better results were obtained using case frames constructed from larger corpora, and the performance showed no saturation even when the corpus size was 1.6 billion sentences.

The findings suggest that increasing corpus size is more essential for acquiring structured knowledge than for acquiring surface statistics of a corpus; and for complex NLP tasks the currently available corpus size is not sufficient.

References

Michaela Atterer and Hinrich Schütze. 2006. The effect of corpus size in combining supervised and unsupervised training for disambiguation. In *Proc. of COLING-ACL'06*, pages 25–32.

Michele Banko and Eric Brill. 2001a. Mitigating the paucity-of-data problem: Exploring the effect of training corpus size on classifier performance for natural language processing. In *Proc. of HLT'01*.

Michele Banko and Eric Brill. 2001b. Scaling to very very large corpora for natural language disambiguation. In *Proc. of ACL'01*, pages 26–33.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proc. of EMNLP-CoNLL'07*, pages 858–867.

Ann Gledson and John Keane. 2008. Using web-search results to measure word-group similarity. In *Proc. of COLING'08*, pages 281–288.

David Graff. 2003. English Gigaword. Technical Report LDC2003T05, Linguistic Data Consortium, Philadelphia, PA USA.

Peter Halacsy, Andras Kornai, Laszlo Nemeth, Andras Rung, Istvan Szakadat, and Vikto Tron. 2004. Creating open language resources for Hungarian. In *Proc. of LREC'04*, pages 203–210.

IREX Committee, editor. 1999. *Proc. of the IREX Workshop*.

Daisuke Kawahara and Sadao Kurohashi. 2006a. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In *Proc. of HLT-NAACL'06*, pages 176–183.

Daisuke Kawahara and Sadao Kurohashi. 2006b. Case frame compilation from the web using high-performance computing. In *Proc. of LREC'06*, pages 1344–1347.

Daisuke Kawahara and Sadao Kurohashi. 2007. Probabilistic coordination disambiguation in a fully-lexicalized Japanese parser. In *Proc. of EMNLP-CoNLL'07*, pages 306–314.

Daisuke Kawahara, Ryohei Sasano, and Sadao Kurohashi. 2004. Toward text understanding: Integrating relevance-tagged corpora and automatically constructed case frames. In *Proc. of LREC'04*, pages 1833–1836.

Adam Kilgarriff and Gregory Grefenstette. 2003. Introduction to the special issue on the web as corpus. *Computational Linguistic*, 29(3):333–347.

Taku Kudo and Hideto Kazawa. 2007. Web Japanese N-gram version 1, published by Gengo Shigen Kyokai.

Mirella Lapata and Frank Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2:1:1–31.

Vinci Liu and James R. Curran. 2006. Web text corpus for natural language processing. In *Proc. of EACL'06*, pages 233–240.

Kikuo Maekawa. 2006. Kotonoha, the corpus development project of the National Institute for Japanese language. In *Proc. of the 13th NIJL International Symposium*, pages 55–62.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proc. of HLT-NAACL'06*, pages 152–159.

Natalia N. Modjeska, Katja Markert, and Malvina Nissim. 2003. Using the web in machine learning for other-anaphora resolution. In *Proc. of EMNLP-2003*, pages 176–183.

Preslav Nakov and Marti Hearst. 2005. A study of using search engine page hits as a proxy for n-gram frequencies. In *Proc. of RANLP'05*.

Preslav Nakov and Marti A. Hearst. 2008. Solving relational similarity problems using the web as a corpus. In *Proc. of ACL-HLT'08*, pages 452–460.

Ryohei Sasano and Sadao Kurohashi. 2008. Japanese named entity recognition using structural natural language processing. In *Proc. of IJCNLP'08*, pages 607–612.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2008. A fully-lexicalized probabilistic model for japanese zero anaphora resolution. In *Proc. of COLING'08*, pages 769–776.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of ACL-HLT'08*, pages 665–673.

The National Language Institute for Japanese Language. 2004. *Bunruigoihyo*. Dainippon Tosho, (In Japanese).

Martin Volk. 2001. Exploiting the WWW as a corpus to resolve PP attachment ambiguities. In *Proc. of the Corpus Linguistics*, pages 601–606.

Semantic-based Estimation of Term Informativeness

Kirill Kireyev
University of Colorado – Boulder
kireyev@colorado.edu

Abstract

The idea that some words carry more semantic content than others, has led to the notion of term specificity, or informativeness. Computational estimation of this quantity is important for various applications such as information retrieval. We propose a new method of computing term specificity, based on modeling the rate of learning of word meaning in Latent Semantic Analysis (LSA). We analyze the performance of this method both qualitatively and quantitatively and demonstrate that it shows excellent performance compared to existing methods on a broad range of tests. We also demonstrate how it can be used to improve existing applications in information retrieval and summarization.

1 Introduction

The idea that some words carry more semantic content than others has been occurring in various literature in linguistics, psychology and computer science for some time. The intuitive notion of *specificity* has long existed before it was formalized; consider, for example, the distinction between the more general word "beverage" and more specific terms "tea", "coffee" and "cocoa" made by Spärck-Jones (1973). Another informal mention of specificity is mentioned by Gorman (1961):

> *A word may be "abstract" and either general or specific, or "concrete" and either general or specific.*

where it is contrasted with another psycholinguistic property of concreteness, which is generally defined as "the extent to which the word's referent can be touched or felt" (Reilly et al., 2007).

The field of information retrieval has attracted greater attention to the computational estimation and applications of term specificity. It has been noted that words with higher specificity, or *information content*, deserve to be weighted more heavily when matching documents with queries, since these words play a greater importance in characterizing what a query or a document is about. By contrast, *stopwords*, words that contribute the least amount of semantic content, are often downweighted or removed altogether (see (Lo et al., 2005), for example).

In addition to IR, term specificity, or informativeness, has been shown useful in other applications, such as Named Entity Tagging (Rennie et al., 2005), creating back-of-the-book glossaries (Csomai et al., 2007), and extractive summarization (Kireyev, 2008).

A related notion of *communication density* has been introduced by Gorman et al. (2003) in team communication analysis, to measure the extent to which a team conveys information in a concise manner, or, in other words, the rate of meaningful discourse, defined by the ratio of *meaningfulness* to number of words spoken. The *meaningfulness* described here should not be confused with psycholinguistic quality of meaningfulness as described by Toglia and Battig (1978), which is the degree to which a word is associated with other words.

In this paper we consider the terms *specificity*, *informativeness* and *information content* of words to mean the same thing. A precise formulation or analysis of important qualitative characteristics of these concepts has not been performed in previous literature; we hope to make some progress in that direction in this paper.

Our main goal is to introduce a new method of computing word specificity based on the rate and strength of semantic associations between words, as modeled by Latent Semantic Analysis (LSA).

2 Previous Approaches

To date, most of the known approaches to estimating term informativeness have relied on frequency-based methods.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 530–538,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

A very basic, yet surprisingly effective approach to measuring term informativeness is its frequency of occurrence in a large representative corpus of language. Spärck Jones (1973) defines *IDF* or *inverse document frequency*, which is determined by the probability of occurrence of documents containing a particular word:

$$IDF(w) = -\log_2(df_w / D)$$

where D is the total number of documents in the corpus. The assumption behind it is that low frequency words tend to be rich in content, and vice versa.

Church and Gale (1995) correctly note that this measure is fundamentally different from c*ollection frequency* f_w, (the total number of times the word type occurs in the corpus) or its transformations, despite the fact that the two measures appear highly correlated. In fact, what is particularly of interest are the words for which these two quantities deviate the most. This happens most dramatically for most informative, or content words, such as "boycott" (Church, 1995a). These words happen to exhibit "bursty" behavior, where they tend to appear multiple times but in fewer documents, thus having $f_w > df_w$. In contrast, less content-loaded words like "somewhat" tend to occur on average once in documents, and thus have similar values for collection and document frequencies ($f_w \approx df_w$). As a result, more informative words can be less accurately estimated by the Poisson distribution, which is based on the simplistic assumption that the expected number of occurrences of word in a document can be estimated by its total number of occurrences in the corpus.

Most prominent statistical measures of term informativeness rely on quantifying this notion of deviation from the Poisson distribution. If the mean expected word rate is:

$$\bar{t}_w = \frac{f_w}{D}$$

then the *variance* metric can be defined as:

$$variance(w) = \frac{1}{D-1}\sum_{d=1}^{D}(t_{dw} - \bar{t}_w)^2$$

where t_{dw} is the actual number of occurrences of term w in document d. The idea is that a higher variance would indicate greater deviation from expected frequency of occurrence in a document, which is assumed to be higher for informative words.

Another measure, suggested by Church and Gale (1995a) is *burstiness* which attempts to compare collection frequency and document frequency directly:

$$burstiness(w) = \frac{\bar{t}_w}{df_w / D} = \frac{f_w}{df_w}$$

Church and Gale also noted that nearly all words have IDF scores that are larger than what one would expect according to an independence-based model such as the Poisson. They note that interesting or informative words tend to have the largest deviations from what would be expected. They thus introduce the notion of *residual IDF* which measures exactly this deviation:

$$residualIDF(w) = IDF(w) + \log_2(1 - e^{-\bar{t}})$$

Papineni (2001) introduces the notion of *gain*:

$$gain(w) = \frac{df_w}{D}\left(\frac{df_w}{D} - 1 - \log(\frac{df_w}{D})\right)$$

This measure tends to give low weights to very high- and very low- frequency words.

Most closely related to our work is the notion of *meaningfulness* in (Gorman et al 2003), computed as the LSA vector length. We will discuss it further in the subsequent sections, and show that a small but crucial modification to this quantity gives the best results.

3 Using Latent Semantic Analysis for Approximating Term Informativeness

3.1 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a language model that represents semantic word meaning as vectors in high-dimensional space. Word vectors are positioned in such a way that semantically-related words vectors point in similar directions or have a smaller angle / higher cosine between them. The representation is derived in an unsupervised manner, by observing occurrence patterns of words in a large corpus of natural language documents. Singular Value Decomposition on the matrix of word/document occurrence counts is used to derive the optimal set of dimensions of the space in which

all of the words can be represented as vectors. The number of dimensions is then artificially reduced to a smaller number (typically around 300) of most important dimensions, which has the effect of smoothing out incidental relationships and preserving significant ones between words.

The resulting geometric space allows for straightforward representation of meaning of words and/or documents; the latter are simply a weighted geometric composition of constituent word vectors. Similarity in meaning between a pair of words or documents can be obtained by computing the cosine between their corresponding vectors. For details of LSA, please see (Landauer et al., 2007), and others

3.2 LSA Term Vector Length

Most of the LSA applications focus on comparing semantic similarity between words and/or text, using the cosine measure of the angle between the corresponding vectors. There is, however, another significant characteristic of LSA word vectors besides their direction in space; it is their vector length. The vector length for words differs significantly, as is shown in Table 1.

Word	df_w	Vector Length
dog	1365	1.3144
green	2067	0.7125
run	2721	0.4788
puppy	127	0.2648
electron	264	0.9009
the	44474	0.0098

Table 1: LSA vector length for some of the words in TASA corpus.

The vector length plays a very important role in many LSA calculations, in particular – in giving relative weights to the word vectors that constitute a particular text passage.

What causes differences in vector lengths? They are based roughly on how much information LSA learns about a word based on its patterns of occurrence in the corpus. Kintsch (2001) writes:

> Intuitively, the vector length tells us how much information LSA has about this vector. [...] Words that LSA knows a lot about (because they appear frequently in the training corpus[...]) have greater vector lengths than words LSA does not know well. Function words that are used frequently in many different contexts have low vector lengths -- LSA knows nothing about them and cannot tell them apart since they appear in all contexts.

Essentially, there are two factors that affect vector length: (1) number of occurrences and (2) the consistency of contexts in which the word occurs.

3.3 Deriving Specificity from Vector Length

Based on the observations above we propose a new metric of term informativeness, or specificity, which we call *LSAspec*, which is simply the ratio of LSA vector length to the number of documents in the LSA training corpus that contain a particular word:

$$LSAspec(w) = \|\vec{w}\| / df_w$$

The value can be interpreted as the rate of vector length growth. We argue that more specific, or informative, words have the greatest rate of vector length growth; LSA learns about their meaning faster, with relatively fewer exposures. To illustrate this concept, let's look at a few examples, that were obtained by controlling the number of occurrences of a particular word in the LSA training corpus. The base corpus was obtained using the 44000-passage TASA corpus with all passages containing the three words below initially removed. Each data point on the graph reflects the vector length of a particular word, after training LSA on the base corpus plus the specified number of passages containing a particular word added back. Highly specific words like "cellulose" gain vector length quite quickly compared to a low-specificity word like "dismay".

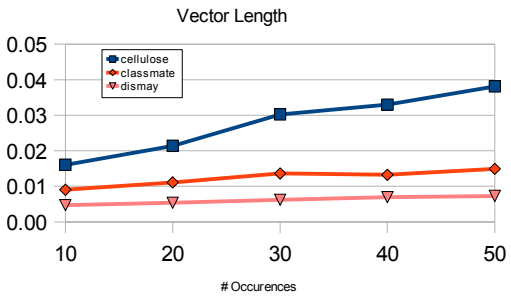


Illustration 1: Vector lengths for some words vs the number of documents containing those words.

4 Comparison of Specificity Metrics

Past attempts to examine the merits of various existing term informativeness estimation methods in the literature thus far has largely involved em-

pirical summative evaluations as part of information retrieval or named entity tagging systems (Rennie et al., 2005). Here, we provide some measures which hopefully provide more illuminating insights into the various methods.

In all of the tests below we derived the metrics (including the LSA space for *LSAspec*) from the same corpus – MetaMetrics 2002 corpus, composed of ~188k passages mostly used in educational texts. No stemming or stopword removal of any kind was performed. All word types were converted to lowercase. We computed the specificity score for each of the 174,374 most frequent words in the corpus using each of the metrics described above: *LSAspec, IDF, residualIDF, burstiness, gain* and *variance*.

4.1 Correlation with Number of Senses

Intuitively, one would expect more specific words to have more precise meaning, and therefore, generally fewer senses. For example, "electron" is a specific physics term that has only one sense, whereas "run" has a very general meaning, and thus has over 50 senses in the WordNet database (Miller et al., 1990). There are many exceptions to this, of course, but overall, one would expect a negative correlation between specificity and number of senses.

In this test, we measure the correlation between the specificity score of a word by various methods and its number of senses in WordNet version 3.0. A total of 75,978 words were considered. We use Spearman correlation coefficient, since the relationships are likely to be non-linear.

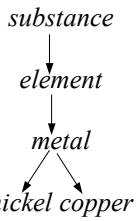
Metric	Corr	Metric	Corr
LSAspec	**-0.46**	burstiness	-0.02
IDF	-0.44	variance	0.40
residualIDF	-0.03	gain	0.44

Table 2: Correlation of specificity metrics with number of senses in WordNet

LSAspec gives the highest negative correlation with number of WordNet senses.

4.2 Correlation with Hypernymy

WordNet organizes concepts into a hypernymy tree, where each parent node is a hypernym of the child node below it. For example:



In general one would expect that for each pair of child-parent pairs in the hypernym tree, the child will have greater specificity than the parent[1]. We examined of a total of 14451 of such hypernym word pairs and computed how often the child's informativeness score, according to each of the measures, is greater than its parent's (its hypernym's) score.

Metric	Percent	Metric	Percent
IDF	**88.8%**	burstiness	47.2%
LSAspec	87.7%	variance	13.4%
residualIDF	48.8%	gain	11.1%

Table 3: Percentage of the time specificity of child exceeds that of its hypernym in WordNet

4.3 Writing Styles and Levels

One may expect that the specificity of words on average would change with texts that are known to be of different writing styles and difficulty level. To test this hypotheses we extracted texts from the TASA collection of educational materials. The texts are annotated with genre ("Science", "Social Studies" or "Language Arts"), and difficulty level on the DRP readability scale (Koslin et al., 1987). Intuitively, one would expect to see two patterns among these texts:

(1) The specificity of words would generally increase with increasing level of difficulty of texts.

(2) Informative (Science) texts should have more specific terms than narrative (Language Arts) texts; with Social Studies somewhere in between (McCarthy et al., 2006).

We extracted 100 text passages for each combination of style ("Science", "Social Studies", "Language Arts") and DRP difficulty level (50, 55, 60, 65, 70)[2], thus resulting in 15 batches of 100 passages. For each passage we computed the median specificity measure of each unique word type in

[1] In practice this is more difficult to determine, since some WordNet entries are actually phrases, rather than words (e.g. "tulip" ← "liliaceous plant" ← ... ← "plant"). In such cases we search up the tree until we stumble upon a node where the entry (or one of the entries) is a single word.

[2] DRP level of 50 roughly corresponds to the beginning of 6th grade in US schools, 70 corresponds to end of 10th grade.

the passage, and averaged these values over 100 passages of each batch. Table 4 shows the results.

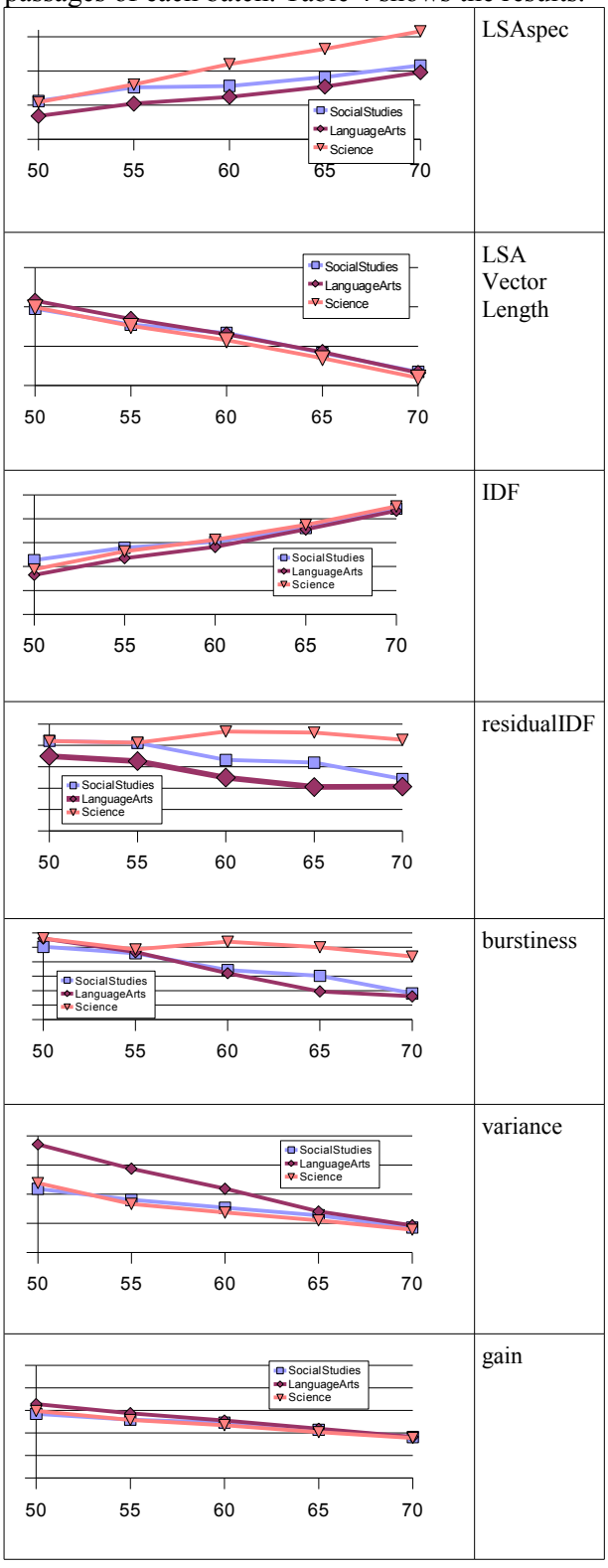


Table 4: Average median specificity scores for texts of different genres and DRP levels.

Note that the absolute values for a particular batch of texts are not important in this case; it's the relative differences between batches of different styles and difficulty levels that are of interest. Of all the measures, only *LSAspec* appears to exhibit the two characteristics described above (increasing with text difficulty, and separating the three genres in the expected way). The metrics *residualIDF* and *burstiness* also appear to separate the genres as expected, but they do not increase with text difficulty.

It is also evident that *LSA Vector Length* alone does not serve as a good measure of informativeness, contrary to its use as such in (Gorman et al., 2003). In fact, it shows the most dramatic and reliable *inverse* relationship with text difficulty. This is likely due to the fact that texts of lower difficulty use common (easier) words more often; these words tend to have longer LSA vector lengths.

4.4 Back-of-the-Book Glossary

Educational textbooks typically have a glossary (index) at the end which lists important terms or concepts mentioned in the book. One would expect these terms to have greater informativeness compared to other words in the textbook. This was a crucial assumption used by Csomai and Mihalcea (2007), who used informativeness (as measured by *IDF* and other metrics) as one of the main features used to automatically generate glossaries from textbooks.

We can use existing textbooks and their glossaries to validate this assumptions, by observing the extent to which the words in the glossary are ranked higher by different specificity metrics compared to other words. Note that the goal here is not to actually achieve optimal performance in automatically finding glossary words; for this reason we do not use recall/precision- based evaluation or rely on additional features such as term frequency (or the popular $tf_w \cdot idf_w$ measure). Rather the goal is to simply see how much the glossary words exhibit the property (informativeness) that we are trying to compute with various methods.

We obtained a collection of textbook chapters (middle-school level material from Prentice Hall Publishing) and their corresponding glossaries, in two different genres: 8 on World Studies (e.g. "Africa", "Medieval Times") and 13 on Science (e.g. "Structure of Animals", "Electricity"). Each

chapter was converted into text and a list of unique words was extracted.

For each of the specificity metrics, we compute how well it predicts glossary words:

1. Compute the specificity of each word in a chapter, according to the metric.
2. Order all the words in decreasing order of specificity.
3. Compute the median percentile rank (position) in the list above of all single-word entries in the glossary (top word has the rank of 0; bottom has a rank of 100).

If a specificity metric predicts the glossary words well, we would expect the average rank to be low; i.e. glossary words would be near the top of the specificity-ordered list.

Metric	Word Studies (~9000 total wds / ch ~260 gloss wds / ch)	Science (~1000 total wds / ch ~20 gloss wds / ch)
LSAspec	**0.21**	**0.10**
residualIDF	0.21	0.11
burstiness	0.21	0.12
IDF	0.29	0.16
variance	0.49	0.64
gain	0.51	0.68

Table 5: Average median rank of glossary words among all other words in textbook by specificity.

LSAspec shows the lowest median percentile for both genres of books.

4.5 Qualitative Analysis

It is useful to inspect the significant differences between the word rankings by different methods, to see if some notable patterns emerge. We can find words on which the methods disagree most dramatically by observing which of them have the most significant differences of position (0-100) in the word lists ranked by different specificity metrics. To avoid dealing with overly-rare words, we restrict our attention to the 23,000 most frequent words in the corpus.

Let's first compare *LSAspec* and *residualIDF*. From the list of 100 words with the most extreme disagreements, we select some examples that have high rank for *LSAspec* (and low for *residuaIDF)* and vice-versa. From Table 6 we can see that *residualIDF* misses some term words (such as "chromatin") which *LSAspec* correctly rates as highly-specific words. Conversely, *residualIDF*, incorrectly ranks common words like "her" and "water" as highly-specific. The reason for this behavior is that words like "chromatin" happen to occur only once per document in the texts they are mentioned (e.g. $df_{cromatin} = tf_{chromatin} = 7$), whereas "her" and "school" tend to occur frequently per document. In real applications "her" will probably be discarded using stopword lists, but "school" will probably not.

Word	LSAspec	residualIDF
oviducts	0.5	98.8
cuspids	0.6	98.8
chromatin	0.7	98.7
disassembly	0.7	98.7
her	99.9	1.5
my	99.9	3.5
water	97.5	5.1
school	97.8	10.3

Table 6: Words ranked most differently by LSAspec and residualIDF

Comparing *LSAspec* and *burstiness* we see almost the same pattern, which is not surprising, since *burstiness* and *residualIDF* work from the same assumptions that content words tend to occur multiple times but in fewer documents.

The table below lists examples of most notable differences between *LSAspec* and *IDF*.

Word	LSAspec	IDF
billy	10.3	93.5
jack	15.0	95.9
melody	4.1	83.8
cells	10.8	86.3
inducing	34.0	9.8
vagueness	32.5	9.6
initiating	31.5	8.7
apathetic	32.3	9.8

Table 7: Words ranked most differently by LSAspec and IDF and their percentiles

There is a large disagreement between rankings of common proper names (e.g. "jack"). It is not clear what the correct answer for these should be, although Rennie & Jaakkola (2005) use informativeness for named entity detection, assuming that proper names should have high specificity. Common but important words like "melody" and "cells" are considered low-specificity by *IDF*. By

contrast, rare but vague words like "inducing" or "vagueness" are improperly given a high specificity ranking.

5 Applications in LSA

Having demonstrated that our word specificity metric performs well with regards to some natural linguistic phenomena, we can now show that it can be used successfully as part of existing NLP technologies. Here we will focus particularly on applications within Latent Semantic Analysis (LSA), although it is highly likely that this specificity metric can be used successfully in other places as well.

We will demonstrate that *LSAspec* shows better results that the conventional term weighting scheme in LSA. It is also important to note that although *LSAspec* is derived using LSA, it is in fact logically independent from the term weighting mechanism used by LSA; other metrics (such as the ones described above) could also be potentially used for LSA term weighting.

In order to represent the meaning of text in LSA, one typically computes the document vector of the text by geometric addition of word vectors for each of the constituent words:

$$\vec{V}_d = \sum_{w \in d} a_w * \log(1 + t_{dw}) * \vec{v}_w$$

where a_w is the log-entropy weight of the word w, typically set to $tf_w \cdot idf_w$ (or some variation thereof) , t_{dw} is the number of occurrences of the word w in the document, and v_w is the vector of the word. Implicit in v_w is its geometric length, which tends to be much greater for frequently-used words (unless they are extremely vague). It is tempered somewhat by a_w which is higher for content words, but perhaps not effectively enough, as the subsequent tests will show. McNamara et al. (2007) experimented with changing the weighting scheme, mainly focusing on prioritizing rare vs. frequent words, and has shown significant differences in short-sentence comparison results.

In the sections below we compare the original LSA weighting scheme with our new scheme based on *LSAspec:*

$$\vec{V}_d = \sum_{w \in d} LSAspec(w) * \log(1 + t_{dw}) * \frac{\vec{v}_w}{||\vec{v}_w||}$$

In other words, we replace the weight parameter a_w and the implicit weight contained in the length of each word vector (by normalizing it) with the specificity value of *LSAspec.*

We show that the resulting term weighting scheme improves performance in three important applications: information retrieval, gisting and short-sentence comparison.

5.1 Information Retrieval

LSA was first introduced as Latent Semantic Indexing (Deerwester et al, 1990), designed for the goal of more effective information retrieval by representing both documents and queries as vectors in a common latent semantic space.

In this IR context, the type of term weighting used to compose document and query vectors plays an important role. We show that using our *LSAspec*-based term weighting gives superior performance to the traditional weighting scheme described in the previous section.

We used the SMART Time[3] dataset, a collection of 425 documents and 83 queries related to Time magazine news articles. For this task only, we used a LSA space that was built using the AQUAINT-2 corpus[4], a large collection (~440,000) of news articles from prominent newspapers such as the New York Times. The variable parameter in the LSA IR models was the cosine threshold between the document and the query, which was varied between 0 and 1

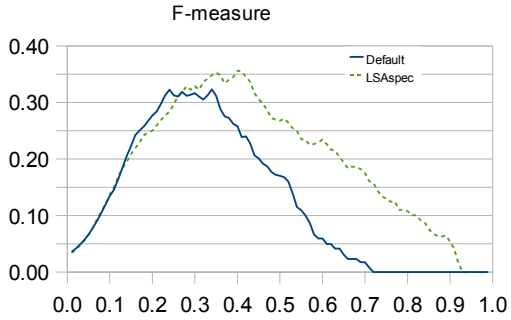


Figure 2: The performance of default LSA and LSA+LSASpec on SMART IR dataset.

Figure 1 shows the performance of the original LSA and LSA with *LSAspec*[5] term weighting method, in terms of the F-measure, which is the harmonic mean of precision and recall; a higher value means better performance. The abscissa in

[3] ftp://ftp.cs.cornell.edu/pub/smart/time/

[4] TREC conference: http://trec.nist.gov/

[5] *LSAspec* measure was the same as before, derived from LSA built on MetaMetrics corpus.

the graph is the value of the threshold cosine parameter. The *LSAspec* term weighting outperforms the original term weighting.

5.2 Sentence Similarity

Here we analyze performance of the two LSA term weighting methods on automated sentence similarity comparisons. Although LSA works best on units of text of paragraph-size or larger, it can work reasonably well on sentence-length units.

We use the dataset reported by McNamara (2007), where the authors collected a set of sentence pairs from several books. A total of 96 sentence pairs was provided, consisting of a combination of subsequent sentences in the book (16), non-adjacent sentences in the same book (16), sentences from two different books (48), and sentences where one is a manually-created paraphrase of one another (16). The standard of reference for this task is human similarity ratings of these sentences within each pair, reported on a Likert scale between 6 (most similar) and 1 (completely dissimilar). Here we report correlations between human rating and LSA similarity with the two term weighting metrics.

Original LSA:	0.66	**LSA + *LSAspec:***	0.85

Using *LSAspec* term weighting gives better performance compared to the original LSA term weighting scheme.

5.3 Gisting (Very Short Summarization)

The ability to represent documents and words in a common geometric space allows LSA to easily compute the gist of a document by finding the word (or sentence) whose vector is most similar by cosine metric to the document vector. This word can be interpreted as the most representative of the cumulative meaning of the document; it can also be thought as a one-word summary of the document. Gisting is discussed from a psychological perspective by Kintsch (2002).

Once again, the choice of term weighting mechanism can make a significant difference in how the overall document vector is constructed. Here, we compare the original weighting scheme and *LSAspec* in the performance on gisting. To perform this evaluation, we selected 46 well-written Wikipedia[6] articles in various categories: Sports, Animals, Countries, Sciences, Religions, Diseases. The original single-word Wikipedia title of each of the articles can be thought as the optimal one-word gist of the article, thus serving as a reference answer in evaluation. A perfect gisting performance by the model would always select the original title as the closest word to the meaning of the document. We also measure the position of the original title in the list of all words in the article ranked by their similarity to the document vector, and ranging from 0 (original title picked as top word) and 1. Table 10 shows a few examples of both the top word and rank of the title, as well as the overall mean rank of all 46 articles.

Title	Orig LSA		LSA + LSAspec	
	top word	rank	top word	rank
Skiing	skiing	0.0000	skiing	0.0000
Thailand	buddhism	0.0189	thailand	0.0000
Sociology	sociologists	0.0012	sociology	0.0000
Pneumonia	infections	0.0064	infections	0.0092
Mean rank (all 46 articles)		**0.0191**		**0.0061**
St. dev. of rank		**0.0847**		**0.0133**

Table 8: Examples of gisting (picking most representative word for text) in with and without LSASpec in LSA

Using *LSAspec* noticeably improves gisting performance, compared to the original LSA term weighting method, as is evidenced by much lower mean rank of the original title.

6 Conclusion

We have introduced a new method of measuring word informativeness. The method gives good results modeling some real linguistic phenomena, and improves LSA applications.

We attempted to look more deeply at the relevant characteristics of word specificity (such as correlation with number of senses). Our method seems to correspond with intuition on emulating a wide range of these characteristics. It also avoids a lot of pitfalls of existing methods that are based purely on frequency statistics, such as unduly prioritizing rare but vague words.

Further research should examine the stability of this method (compared to others) with regards to variation/size of the training corpus. It should also analyze application of the method in other natural language tasks. Lastly, it should be correlated with human judgments, similar to other psycholinguistic properties.

[6] http://en.wikipedia.org, circa May 2008.

References

Kenneth W. Church and William A. Gale. 1995. Poisson mixtures. *Journal of Natural Language Engineering*, 1995

Kenneth W. Church and William A. Gale. 1995a. Inverse document frequency (IDF): A measure of deviation from Poisson. *In Proceedings of the Third Workshop on Very Large Corpora*, pp 121–130, 1995.

András Csomai and Rada Mihalcea. 2007. Investigations in Unsupervised Back-of-the-Book Indexing. In *Proceedings of the Florida Artificial Intelligence Research Society*, Key West.

Scott Deerwester, Susan T. Dumais, George W. Furnas and Thomas K. Landauer. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41.

Aloysia M. Gorman. 1961. Recognition Memory for Nouns as a Function of Abstractness and Frequency. *Journal of Experimental Psychology*. Vol. 61, No. 1.

Jamie C. Gorman, Peter W. Foltz, Preston A. Kiekel and Melanie J. Martin. 2003. Evaluation of Latent Semantic Analysis-based Measures of Team Communication Content. *Proceedings of the Human Factors and Ergonomics Society, 47th Annual Meeting*, pp 424-428.

Walter Kintsch. 2002. On the notions of theme and topic in psychological process models of text comprehension. In M. Louwerse & W. van Peer (Eds.) *Thematics : Interdisciplinary Studies*, Amsterdam, Benjamins, pp 157-170.

Walter Kintsch. 2001. Predication. *Journal of Cognitive Science*, 25.

Kirill Kireyev. 2008. Using Latent Semantic Analysis for Extractive Summarization. *Proceedings of Text Analysis Conference*, 2008.

B. L. Koslin, S. Zeno, and S. Koslin. 1987. The DRP: An Effective Measure in Reading. *New York College Entrance Examination Board.*

Thomas K Landauer and Susan Dumais. 1997. A solution to Plato's problem: The Latent Semantic nalysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review,* 104, pp 211-240.

Thomas K Landauer, Danielle S. McNamara, Simon Dennis, and Walter Kintsch. 2007. *Handbook of Latent Semantic Analysis* Lawrence Erlbaum.

Rachel TszWai Lo, Ben He, and Iadh Ounis. 2005. Automatically Building a Stopword List for an Information Retrieval System. *5th Dutch-Belgium Information Retrieval Workshop (DIR)*. 2005.

Philip M. McCarthy, Arthur C. Graesser, Danielle S. McNamara. 2006. Distinguishing Genre Using CohMetrix Indices of Cohesion. 16th *Annual Meeting of the Society for Text and Discourse, Minneapolis, MN, 2006.*

Danielle S. McNamara, Zhiqiang Cai, and MaxM. Louwerse. 2007. Optimizing LSA Measures of Cohesion. *Handbook of Latent Semantic Analysis* . Mahwah, NJ: Erlbaum. ch 19, pp 379-399.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross and Katherine Miller. 1990. WordNet: An on-line lexical database. *International Journal of Lexicography,* 3 (4), 1990.

Kishore Papineni. 2001. Why inverse document frequency. In *Proceedings of the NAACL*, 2001

Jamie Reilly and Jacob Kean. 2007. Formal Distinctiveness Of High- and Low- Imageability Nouns: Analyses and Theoretical Implications. *Cognitive Science*, 31.

Jason D. M. Rennie and Tommi Jaakkola. 2005. Using Term Informativeness for Named Entity Detection. *Proceedings of ACM SIGIR* 2005.

Karen Spärck-Jones. 1973. "A Statistical Interpretation of Term Specificity and its Application in Retrieval," *Journal of Documentation*, 28:1.

Michael P. Toglia and William R. Battig. 1978. *Handbook of semantic word norms*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Optimal Reduction of Rule Length in Linear Context-Free Rewriting Systems

Carlos Gómez-Rodríguez[1], **Marco Kuhlmann**[2], **Giorgio Satta**[3] **and David Weir**[4]

[1] Departamento de Computación, Universidade da Coruña, Spain (cgomezr@udc.es)

[2] Department of Linguistics and Philology, Uppsala University, Sweden (marco.kuhlmann@lingfil.uu.se)

[3] Department of Information Engineering, University of Padua, Italy (satta@dei.unipd.it)

[4] Department of Informatics, University of Sussex, United Kingdom (davidw@sussex.ac.uk)

Abstract

Linear Context-free Rewriting Systems (LCFRS) is an expressive grammar formalism with applications in syntax-based machine translation. The parsing complexity of an LCFRS is exponential in both the *rank* of a production, defined as the number of nonterminals on its right-hand side, and a measure for the discontinuity of a phrase, called *fan-out*. In this paper, we present an algorithm that transforms an LCFRS into a strongly equivalent form in which all productions have rank at most 2, and has minimal fan-out. Our results generalize previous work on Synchronous Context-Free Grammar, and are particularly relevant for machine translation from or to languages that require syntactic analyses with discontinuous constituents.

1 Introduction

There is currently considerable interest in syntax-based models for statistical machine translation that are based on the extraction of a synchronous grammar from a corpus of word-aligned parallel texts; see for instance Chiang (2007) and the references therein. One practical problem with this approach, apart from the sheer number of the rules that result from the extraction procedure, is that the parsing complexity of all synchronous formalisms that we are aware of is exponential in the **rank** of a rule, defined as the number of nonterminals on the right-hand side. Therefore, it is important that the rules of the extracted grammar are transformed so as to minimise this quantity. Not only is this beneficial in terms of parsing complexity, but smaller rules can also improve a translation model's ability to generalize to new data (Zhang et al., 2006).

Optimal algorithms exist for minimising the size of rules in a Synchronous Context-Free Grammar (SCFG) (Uno and Yagiura, 2000; Zhang et al., 2008). However, the SCFG formalism is limited to modelling word-to-word alignments in which a single continuous phrase in the source language is aligned with a single continuous phrase in the target language; as defined below, this amounts to saying that SCFG have a **fan-out** of 2. This restriction appears to render SCFG empirically inadequate. In particular, Wellington et al. (2006) find that the coverage of a translation model can increase dramatically when one allows a bilingual phrase to stretch out over three rather than two continuous substrings. This observation is in line with empirical studies in the context of dependency parsing, where the need for formalisms with higher fan-out has been observed even in standard, single language texts (Kuhlmann and Nivre, 2006).

In this paper, we present an algorithm that computes optimal decompositions of rules in the formalism of Linear Context-Free Rewriting Systems (LCFRS) (Vijay-Shanker et al., 1987). LCFRS was originally introduced as a generalization of several so-called mildly context-sensitive grammar formalisms. In the context of machine translation, LCFRS is an interesting generalization of SCFG because it does not restrict the fan-out to 2, allowing productions with *arbitrary* fan-out (and arbitrary rank). Given an LCFRS, our algorithm computes a strongly equivalent grammar with rank 2 and min-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 539–547,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

imal increase in fan-out.[1] In this context, strong equivalence means that the derivations of the original grammar can be reconstructed using some simple homomorphism (c.f. Nijholt, 1980). Our contribution is significant because the existing algorithms for decomposing SCFG, based on Uno and Yagiura (2000), cannot be applied to LCFRS, as they rely on the crucial property that components of biphrases are strictly separated in the generated string: Given a pair of synchronized nonterminal symbols, the material derived from the source nonterminal must precede the material derived from the target nonterminal, or vice versa. The problem that we solve has been previously addressed by Melamed et al. (2004), but in contrast to our result, their algorithm does not guarantee an optimal (minimal) increase in the fan-out of the resulting grammar. However, this is essential for the practical applicability of the transformed grammar, as the parsing complexity of LCFRS is exponential in *both* the rank and the fan-out.

Structure of the paper The remainder of the paper is structured as follows. Section 2 introduces the terminology and notation that we use for LCFRS. In Section 3, we present the technical background of our algorithm; the algorithm itself is discussed in Section 4. Section 5 concludes the paper by discussing related work and open problems.

General notation The set of non-negative integers is denoted by $\mathbb{N}$. For $i, j \in \mathbb{N}$, we write $[i, j]$ to denote the interval $\{ k \in \mathbb{N} \mid i \leq k \leq j \}$, and use $[i]$ as a shorthand for $[1, i]$. Given an alphabet V, we write V^* for the set of all (finite) strings over V.

2 Preliminaries

We briefly summarize the terminology and notation that we adopt for LCFRS; for detailed definitions, see Vijay-Shanker et al. (1987).

2.1 Linear, non-erasing functions

Let V be an alphabet. For natural numbers $r \geq 0$ and $f, f_1, \ldots, f_r \geq 1$, a function

$$g : (V^*)^{f_1} \times \cdots \times (V^*)^{f_r} \to (V^*)^f$$

is called a **linear, non-erasing function** over V of type $f_1 \times \cdots \times f_r \to f$, if it can be defined by an equation of the form

$$g(\langle x_{1,1}, \ldots, x_{1,f_1} \rangle, \ldots, \langle x_{r,1}, \ldots, x_{r,f_r} \rangle) \;=\; \beta_g \,,$$

where $\beta_g = \langle \alpha_{g,1}, \ldots, \alpha_{g,f} \rangle$ is an f-tuple of strings over the variables on the left-hand side of the equation and symbols in V that contains exactly one occurrence of each variable. We call the value r the **rank** of g, the value f its **fan-out**, and write $\rho(g)$ and $\varphi(g)$, respectively, to denote these quantities. Note that, if we assume the variables on the left-hand side of the defining equation of g to be named according to the specific schema given above, then g is uniquely determined by β_g.

2.2 Linear context-free rewriting systems

A **linear context-free rewriting system** (LCFRS) is a construct $G = (V_N, V_T, P, S)$, where: V_N is an alphabet of nonterminal symbols in which each symbol $A \in V_N$ is associated with a value $\varphi(A)$, called its **fan-out**; V_T is an alphabet of terminal symbols; $S \in N$ is a distinguished start symbol with $\varphi(S) = 1$; and P is a set of productions of the form

$$p : A \to g(B_1, B_2, \ldots, B_r) \,,$$

where $A, B_1, \ldots, B_r \in V_N$, and g is a linear, non-erasing function over the terminal alphabet V_T of type $\varphi(B_1) \times \cdots \times \varphi(B_r) \to \varphi(A)$. In a derivation of an LCFRS, the production p can be used to transform a sequence of r tuples of strings, generated by the nonterminals $B_1, \ldots, B_r$, into a single $\varphi(A)$-tuple of strings, associated with the nonterminal A. The values $\rho(g)$ and $\varphi(g)$ are called the **rank** and **fan-out** of p, respectively, and we write $\rho(p)$ and $\varphi(p)$, respectively, to denote these quantities. The rank and fan-out of G, written $\rho(G)$ and $\varphi(G)$, respectively, are the maximum rank and fan-out among all of its productions. Given that $\varphi(S) = 1$, a derivation will associate S with a set of one-component tuples of strings over V_T; this forms the string language generated by G.

Example 1 The following LCFRS generates the string language $\{ a^n b^n c^n d^n \mid n \in \mathbb{N} \}$. We only specify the set of productions; the remaining com-

[1] Rambow and Satta (1999) show that without increasing fan-out it is not always possible to produce even *weakly* equivalent grammars.

ponents of the grammar are obvious from that.

$$\begin{aligned} S \to g_1(R) \quad & g_1(\langle x_{1,1}, x_{1,2}\rangle) = \langle x_{1,1}x_{1,2}\rangle \\ R \to g_2(R) \quad & g_2(\langle x_{1,1}, x_{1,2}\rangle) = \langle ax_{1,1}b, cx_{1,2}d\rangle \\ R \to g_3 \quad & g_3 = \langle \varepsilon, \varepsilon\rangle \end{aligned}$$

The functions g_1 and g_2 have rank 1; the function g_3 has rank 0. The functions g_2 and g_3 have fan-out 2; the function g_1 has fan-out 1. □

3 Technical background

The general idea behind our algorithm is to replace each production of an LCFRS with a set of "shorter" productions that jointly are equivalent to the original production. Before formalizing this idea, we first introduce a specialized representation for the productions of an LCFRS.

We distinguish between occurrences of symbols within a string by exploiting two different notations. Let $\alpha = a_1 a_2 \cdots a_n$ be a string. The occurrence a_i in α can be denoted by means of its **position** index $i \in [n]$, or else by means of its two (left and right) **endpoints**, $i-1$ and i; here, the left (right) endpoint denotes a boundary between occurrence a_i and the previous (subsequent) occurrence, or the beginning (end) of the string α. Similarly, a substring $a_i \cdots a_j$ of α with $i \leq j$ can be denoted by the positions $i, i+1, \ldots, j$ of its occurrences, or else by means of its left and right endpoints, $i-1$ and j.

3.1 Production representation

For the remainder of this section, let us fix an LCFRS $G = (V_N, V_T, P, S)$ and a production $p : A \to g(B_1, \ldots, B_r)$ of G, with g defined as in Section 2.1. We define

$$|p| = \varphi(g) + \sum_{i=1}^{\varphi(g)} |\alpha_{g,i}|.$$

Let \$ be a fresh symbol that does not occur in G. We define the **characteristic string** of the production p as

$$\sigma(p) = \alpha_{g,1}\$ \cdots \$\alpha_{g,\varphi(g)}\,,$$

and the **variable string** of p as the string $\sigma_N(p)$ obtained from $\sigma(p)$ by removing all the occurrences of symbols in V_T.

Example 2 We will illustrate the concepts introduced in this section using the concrete production $p_0 : A \to g(B_1, B_2, B_3)$, where

$$\beta_g \;=\; \langle x_{1,1}ax_{2,1}x_{1,2}, x_{3,1}bx_{3,2}\rangle\,.$$

In this case, we have

$$\begin{aligned} \sigma(p_0) \;&=\; x_{1,1}ax_{2,1}x_{1,2}\$x_{3,1}bx_{3,2}\,, \quad \text{and} \\ \sigma_N(p_0) \;&=\; x_{1,1}x_{2,1}x_{1,2}\$x_{3,1}x_{3,2}\,. \end{aligned}$$ □

Let I be an index set, $I \subseteq [r]$. Consider the set $\mathcal{B}$ of occurrences B_i in the right-hand side of p such that $i \in I$.[2] We define the **position set** of $\mathcal{B}$, denoted by $\Pi_{\mathcal{B}}$, as the set of all positions $1 \leq j \leq |\sigma_N(p)|$ such that the jth symbol in $\sigma_N(p)$ is a variable of the form $x_{i,h}$, for $i \in I$ and some $h \geq 1$.

Example 3 Some position sets of p_0 are

$$\Pi_{\{B_1\}} = \{1,3\}\,, \Pi_{\{B_2\}} = \{2\}\,, \Pi_{\{B_3\}} = \{5,6\}\,.$$ □

A position set $\Pi_{\mathcal{B}}$ can be uniquely expressed as the union of $f \geq 1$ intervals $[l_1+1, r_1], \ldots, [l_f+1, r_f]$ such that $r_{i-1} < l_i$ for every $1 < i \leq f$. Thus we define the set of **endpoints** of $\Pi_{\mathcal{B}}$ as

$$\Delta_{\mathcal{B}} \;=\; \{\, l_j \mid j \in [f]\,\} \cup \{\, r_j \mid j \in [f]\,\}\,.$$

The quantity f is called the **fan-out** of $\Pi_{\mathcal{B}}$, written $\varphi(\Pi_{\mathcal{B}})$. Notice that the fan-out of a position set $\Pi_{\{B\}}$ does not necessarily coincide with the fan-out of the non-terminal B in the underlying LCFRS. A set with $2f$ endpoints always corresponds to a position set of fan-out f.

Example 4 For our running example, we have $\Delta_{\{B_1\}} = \{0,1,2,3\}$, $\Delta_{\{B_2\}} = \{1,2\}$, $\Delta_{\{B_3\}} = \{4,6\}$. Consequently, the fan-out of $\Delta_{\{B_1\}}$ is 2, and the fan-out of $\Delta_{\{B_2\}}$ and $\Delta_{\{B_3\}}$ is 1. Notice that the fan-out of the non-terminal B_3 is 2. □

We drop $\mathcal{B}$ from $\Pi_{\mathcal{B}}$ and $\Delta_{\mathcal{B}}$ whenever this set is understood from the context or it is not relevant. Given a set of endpoints $\Delta = \{i_1, \ldots, i_{2f}\}$ with $i_1 < \cdots < i_{2f}$, we obtain its corresponding position set by calculating the **closure** of Δ, defined as

$$[\Delta] \;=\; \textstyle\bigcup_{j=1}^{f} [i_{2j-1}+1, i_{2j}]\,.$$

[2] To avoid clutter in our examples, we abuse the notation by not making an explicit distinction between nonterminals and occurrences of nonterminals in productions.

3.2 Reductions

Assume that $r > 2$. The **reduction** of p by the nonterminal occurrences B_{r-1}, B_r is the ordered pair of productions (p_1, p_2) that is defined as follows. Let $\gamma_1, \ldots, \gamma_n$ be the maximal substrings of $\sigma(p)$ that contain only variables $x_{i,j}$ with $r-1 \leq i \leq r$ and terminal symbols, and at least one variable. Then

$$p_1 : A \to g_1(B_1, \ldots, B_{r-2}, X) \qquad \text{and}$$
$$p_2 : X \to g_2(B_{r-1}, B_r)\,,$$

where X is a fresh nonterminal symbol, the characteristic string $\sigma(p_1)$ is the string obtained from $\sigma(p)$ by replacing each substring γ_i by the variable $x_{r-1,i}$, and the characteristic string $\sigma(p_2)$ is the string $\gamma_1 \$ \cdots \$ \gamma_n$.

Note that the defining equations of neither g_1 nor g_2 are in the specific form discussed in Section 2.1; however, they can be brought into this form by a consistent renaming of the variables. We will silently assume this renaming to take place.

Example 5 The reduction of p_0 by the nonterminal occurrences B_2 and B_3 has $p_1 : A \to g_1(B_1, X)$ and $p_2 : X \to g_2(B_2, B_3)$ with

$$\sigma(p_1) = x_{1,1}x_{2,1}x_{1,2}\$x_{2,2}$$
$$\sigma(p_2) = ax_{2,1}\$x_{3,1}bx_{3,2}$$

or, after renaming and in standard notation,

$$g_1(\langle x_{1,1}, x_{1,2}\rangle, \langle x_{2,1}, x_{2,2}\rangle) = \langle x_{1,1}x_{2,1}x_{1,2}, x_{2,2}\rangle$$
$$g_2(\langle x_{1,1}\rangle, \langle x_{2,1}, x_{2,2}\rangle) = \langle ax_{1,1}, x_{2,1}bx_{2,2}\rangle\,. \square$$

It is easy to check that a reduction provides us with a pair of productions that are equivalent to the original production p, in terms of generative capacity, since

$$g_1(B_1, \ldots, B_{r-2}, g_2(B_{r-1}, B_r)) = g(B_1, \ldots, B_r)$$

for all tuples of strings generated from the nonterminals $B_1, \ldots, B_r$, respectively. Note also that the fan-out of production p_1 equals the fan-out of p. However, the fan-out of p_2 (the value n) may be greater than the fan-out of p, depending on the way variables are arranged in $\sigma(p)$. Thus, a reduction does not necessarily preserve the fan-out of the original production. In the worst case, the fan-out of p_2 can be as large as $\varphi(B_{r-1}) + \varphi(B_r)$.

1: **Function** NAIVE-BINARIZATION(p)
2: result $\leftarrow \emptyset$;
3: currentProd $\leftarrow p$;
4: **while** ρ(currentProd) > 2 **do**
5: $\quad(p_1, p_2) \leftarrow$ any reduction of currentProd;
6: $\quad$result $\leftarrow$ result $\cup\, p_2$;
7: $\quad$currentProd $\leftarrow p_1$;
8: **return** result $\cup$ currentProd;

Figure 1: The naive algorithm

We have defined reductions only for the last two occurrences of nonterminals in the right-hand side of a production p. However, it is easy to see that we can also define the concept for two arbitrary (not necessarily adjacent) occurrences of nonterminals, at the cost of making the notation more complicated.

4 The algorithm

Let G be an LCFRS with $\varphi(G) = f$ and $\rho(G) = r$, and let $f' \geq f$ be a target fan-out. We will now present an algorithm that computes an equivalent LCFRS G' of fan-out at most f' whose rank is at most 2, if such an LCFRS exists in the first place. The algorithm works by exhaustively reducing all productions in G.

4.1 Naive algorithm

Given an LCFRS production p, a naive algorithm to compute an equivalent set of productions whose rank is at most 2 is given in Figure 1. By applying this algorithm to all the productions in the LCFRS G, we can obtain an equivalent LCFRS with rank 2. We will call such an LCFRS a **binarization** of G.

The fan-out of the obtained LCFRS will depend on the nonterminals that we choose for the reductions in line 5. It is not difficult to see that, in the worst case, the resulting fan-out can be as high as $\lceil \frac{r}{2} \rceil \cdot f$. This occurs when we choose $\lceil \frac{r}{2} \rceil$ nonterminals with fan-out f that have associated variables in the string $\sigma_N(p)$ that do not occur at consecutive positions.

The algorithm that we develop in Section 4.3 improves on the naive algorithm in that it can be exploited to find a sequence of reductions that results in a binarization of G that is optimal, i.e., leads to

an LCFRS with *minimal* fan-out. The algorithm is based on a technical concept called *adjacency*.

4.2 Adjacency

Let p be some production in the LCFRS G, and let Δ_1, Δ_2 be sets of endpoints, associated with some sets of nonterminal occurrences in p. We say that Δ_1 and Δ_2 **overlap** if the intersection of their closures is nonempty, that is, if $[\Delta_1] \cap [\Delta_2] \neq \emptyset$. Overlapping holds if and only if the associated sets of nonterminal occurrences are not disjoint. If Δ_1 and Δ_2 do not overlap, we define their **merge** as

$$\oplus(\Delta_1, \Delta_2) \;=\; (\Delta_1 \cup \Delta_2) \setminus (\Delta_1 \cap \Delta_2)\,.$$

It is easy to see that $[\oplus(\Delta_1, \Delta_2)] = [\Delta_1] \cup [\Delta_2]$. We say that Δ_1 and Δ_2 are **adjacent** for a given fan-out f, written $\Delta_1 \leftrightarrow_f \Delta_2$, if Δ_1 and Δ_2 do not overlap, and $\varphi([\oplus(\Delta_1, \Delta_2)]) \leq f$.

Example 6 For the production p_0 from Example 2, we have $\oplus(\Delta_{\{B_1\}}, \Delta_{\{B_2\}}) = \{0, 3\}$, showing that $\Delta_{\{B_1\}} \leftrightarrow_1 \Delta_{\{B_2\}}$. Similarly, we have

$$\oplus(\Delta_{\{B_1\}}, \Delta_{\{B_3\}}) = \{0, 1, 2, 3, 4, 6\}\,,$$

showing that $\Delta_{\{B_1\}} \leftrightarrow_3 \Delta_{\{B_3\}}$, but that neither $\Delta_{\{B_1\}} \leftrightarrow_2 \Delta_{\{B_3\}}$ nor $\Delta_{\{B_1\}} \leftrightarrow_1 \Delta_{\{B_3\}}$ holds. □

4.3 Bounded binarization algorithm

The adjacency-based binarization algorithm is given in Figure 2. It starts with a working set containing the endpoint sets corresponding to each nonterminal occurrence in the input production p. Reductions of p are only explored for nonterminal occurrences whose endpoint sets are adjacent for the target fan-out f', since reductions not meeting this constraint would produce productions with fan-out greater than f'. Each reduction explored by the algorithm produces a new endpoint set, associated to the fresh nonterminal that it introduces, and this new endpoint set is added to the working set and potentially used in further reductions.

From the definition of the adjacency relation $\leftrightarrow_f$, it follows that at lines 9 and 10 of BOUNDED-BINARIZATION we only pick up reductions for p that do not exceed the fan-out bound of f'. This implies soundness for our algorithm. Completeness means that the algorithm fails only if there exists no binarization for p of fan-out not greater than f'. This

```
1:  Function BOUNDED-BINARIZATION(p, f′)
2:  workingSet ← ∅;
3:  agenda ← ∅;
4:  for all i from 1 to ρ(p) do
5:      workingSet ← workingSet ∪ {Δ_{B_i}};
6:      agenda ← agenda ∪ {Δ_{B_i}};
7:  while agenda ≠ ∅ do
8:      Δ ← pop some endpoint set from agenda;
9:      for all Δ_1 ∈ workingSet with Δ_1 ↔_{f′} Δ do
10:         Δ_2 = ⊕(Δ, Δ_1);
11:         if Δ_2 ∉ workingSet then
12:             workingSet ← workingSet ∪ {Δ_2};
13:             agenda ← agenda ∪ {Δ_2};
14: if Δ_{B_1,B_2,...,B_{ρ(p))}} ∈ workingSet then
15:     return true;
16: else
17:     return false;
```

Figure 2: Algorithm to compute a bounded binarization

property is intuitive if one observes that our algorithm is a specialization of standard algorithms for the computation of the closure of binary relations. A formal proof of this fact is rather long and tedious, and will not be reported here. We notice that there is a very close similarity between algorithm BOUNDED-BINARIZATION and the deduction procedure proposed by Shieber et al. (1995) for parsing. We discuss this more at length in Section 5.

Note that we have expressed the algorithm as a decision function that will return true if there exists a binarization of p with fan-out not greater than f', and false otherwise. However, the algorithm can easily be modified to return a reduction producing such a binarization, by adding to each endpoint set $\Delta \in$ workingSet two pointers to the adjacent endpoint sets that were used to obtain it. If the algorithm is successful, the tree obtained by following these pointers from the final endpoint set $\Delta_{\{B_1,\ldots,B_{\rho(p)}\}} \in$ workingSet gives us a tree of reductions that will produce a binarization of p with fan-out not greater than f', where each node labeled with the set $\Delta_{\{B_i\}}$ corresponds to the nonterminal B_i, and nodes labeled with other endpoint sets correspond to the fresh nonterminals created by the reductions.

4.4 Implementation

In order to implement BOUNDED-BINARIZATION, we can represent endpoint sets in a canonical way as $2f'$-tuples of integer positions in ascending order, and with some special null value used to fill positions for endpoint sets with fan-out strictly smaller than f'. We will assume that the concrete null value is larger than any other integer.

We also need to provide some appropriate representation for the set workingSet, in order to guarantee efficient performance for the membership test and the insertion operation. Both operations can be implemented in constant time if we represent workingSet as an $(2 \times f')$-dimensional table with Boolean entries. Each dimension is indexed by values in $[0, n]$ plus our special null value; here n is the length of the string $\sigma_N(p)$, and thus $n = \mathcal{O}(|p|)$. However, this has the disadvantage of using space $\Theta(n^{2f'})$, even in case workingSet is sparse, and is affordable only for quite small values of f'. Alternatively, we can more compactly represent workingSet as a trie data structure. This representation has size certainly smaller than $2f' \times q$, where q is the size of the set workingSet. However, both membership and insertion operations take now an amount of time $\mathcal{O}(2f')$.

We now analyse the time complexity of algorithm BOUNDED-BINARIZATION for inputs p and f'. We first focus on the while-loop at lines 7 to 13. As already observed, the number of possible endpoint sets is bounded by $\mathcal{O}(n^{2f'})$. Furthermore, because of the test at line 11, no endpoint set is ever inserted into the agenda variable more than once in a single run of the algorithm. We then conclude that our while-loop cycles a number of times $\mathcal{O}(n^{2f'})$.

We now focus on the choice of the endpoint set Δ_1 in the inner for-loop at lines 9 to 13. Let us fix Δ as in line 8. It is not difficult to see that any Δ_1 with $\Delta_1 \leftrightarrow_{f'} \Delta$ must satisfy

$$\varphi(\Delta) + \varphi(\Delta_1) - |\Delta \cap \Delta_1| \quad \leq \quad f'. \qquad (1)$$

Let $I \subseteq \Delta$, and consider all endpoint sets Δ_1 with $\Delta \cap \Delta_1 = I$. Given (1), we also have

$$\varphi(\Delta_1) \quad \leq \quad f' + |I| - \varphi(\Delta). \qquad (2)$$

This means that, for each Δ coming out of the agenda, at line 9 we can choose all endpoint sets Δ_1 such that $\Delta_1 \leftrightarrow_{f'} \Delta$ by performing the following steps:

- arbitrarily choose a set $I \subseteq \Delta$;
- choose endpoints in set $\Delta_1 \backslash I$ subject to (2);
- test whether Δ_1 belongs to workingSet and whether Δ, Δ_1 do not overlap.

We claim that, in the above steps, the number of involved endpoints does not exceed $3f'$. To see this, we observe that from (2) we can derive $|I| \geq \varphi(\Delta) + \varphi(\Delta_1) - f'$. The total number of (distinct) endpoints in a single iteration step is $e = 2\varphi(\Delta) + 2\varphi(\Delta_1) - |I|$. Combining with the above inequality we have

$$\begin{aligned} e \quad &\leq \quad 2\varphi(\Delta) + 2\varphi(\Delta_1) - \varphi(\Delta) - \varphi(\Delta_1) + f' \\ &= \quad \varphi(\Delta) + \varphi(\Delta_1) + f' \ \leq \ 3f' \,, \end{aligned}$$

as claimed. Since each endpoint takes values in the set $[0, n]$, we have a total of $\mathcal{O}(n^{3f'})$ different choices. For each such choice, we need to classify an endpoint as belonging to either $\Delta \backslash I$, $\Delta_1 \backslash I$, or I. This amounts to an additional $\mathcal{O}(3^{3f'})$ different choices. Overall, we have a total number of $\mathcal{O}((3n)^{3f'})$ different choices. For each such choice, the test for membership in workingSet for Δ_1 takes constant time in case we use a multi-dimensional table, or else $\mathcal{O}(|p|)$ in case we use a trie. The adjacency test and the merge operations can easily be carried out in time $\mathcal{O}(|p|)$.

Putting all of the above observations together, and using the already observed fact that $n = \mathcal{O}(|p|)$, we can conclude that the total amount of time required by the while-loop at lines 7 to 13 is bounded by $\mathcal{O}(|p| \cdot (3|p|)^{3f'})$, both under the assumption that workingSet is represented as a multi-dimensional table or as a trie. This is also a bound on the running time of the whole algorithm.

4.5 Minimal binarization of a complete LCFRS

The algorithm defined in Section 4.3 can be used to binarize an LCFRS in such a way that each rule in the resulting binarization has the minimum possible fan-out. This can be done by applying the BOUNDED-BINARIZATION algorithm to each production p, until we find the minimum value for the

1: **Function** MINIMAL-BINARIZATION(G)
2: $p_b = \emptyset$ {Set of binarized productions}
3: **for all** production p of G **do**
4: f' = *fan-out*(p);
5: **while** not BOUNDED-BINARIZATION(p, f') **do**
6: $f' = f' + 1$;
7: add result of BOUNDED-BINARIZATION(p, f') to p_b; {We obtain the tree from BOUNDED-BINARIZATION as explained in Section 4.3 and use it to binarize p}
8: **return** p_b;

Figure 3: Minimal binarization by sequential search

bound f' for which this algorithm finds a binarization. For a production with rank r and fan-out f, we know that this optimal value of f' must be in the interval $[f, \lceil \frac{r}{2} \rceil \cdot f]$ because binarizing a production cannot reduce its fan-out, and the NAIVE-BINARIZATION algorithm seen in Section 4.1 can binarize any production by increasing fan-out to $\lceil \frac{r}{2} \rceil \cdot f$ in the worst case.

The simplest way of finding out the optimal value of f' for each production is by a sequential search starting with $\varphi(p)$ and going upwards, as in the algorithm in Figure 3. Note that the upper bound $\lceil \frac{r}{2} \rceil \cdot f$ that we have given for f' guarantees that the while-loop in this algorithm always terminates.

In the worst case, we may need $f \cdot (\lceil \frac{r}{2} \rceil - 1) + 1$ executions of the BOUNDED-BINARIZATION algorithm to find the optimal binarization of a production in G. This complexity can be reduced by changing the strategy to search for the optimal f': for example, we can perform a binary search within the interval $[f, \lceil \frac{r}{2} \rceil \cdot f]$, which lets us find the optimal binarization in $\lfloor \log(f \cdot (\lceil \frac{r}{2} \rceil - 1) + 1) \rfloor + 1$ executions of BOUNDED-BINARIZATION. However, this will not result in a practical improvement, since BOUNDED-BINARIZATION is exponential in the value of f' and the binary search will require us to run it on values of f' larger than the optimal in most cases. An intermediate strategy between the two is to apply exponential backoff to try the sequence of values $f-1+2^i$ (for $i = 0, 1, 2 \ldots$). When we find the first i such that BOUNDED-BINARIZATION does not fail, if $i > 0$, we apply the same strategy to the interval $[f-1+2^{i-1}, f-2+2^i]$, and we repeat this method to shrink the interval until BOUNDED-BINARIZATION does not fail for $i = 0$, giving us our optimal f'. With this strategy, the amount of executions of the algorithm that we need in the worst case is

$$\frac{1}{2}(\lceil \log(\omega) \rceil + \lceil \log(\omega) \rceil^2) + 1 \,,$$

where $\omega = f \cdot (\lceil \frac{r}{2} \rceil - 1) + 1$, but we avoid using unnecessarily large values of f'.

5 Discussion

To conclude this paper, we now discuss a number of aspects of the results that we have presented, including various other pieces of research that are particularly relevant to this paper.

5.1 The tradeoff between rank and fan-out

The algorithm introduced in this paper can be used to transform an LCFRS into an equivalent form with rank 2. This will result into a more efficiently parsable LCFRS, since rank exponentially affects parsing complexity. However, we must take into account that parsing complexity is also influenced by fan-out. Our algorithm guarantees a minimal increase in fan-out. In practical cases it seems such an increase is quite small. For example, in the context of dependency parsing, both Gómez-Rodríguez et al. (2009) and Kuhlmann and Satta (2009) show that all the structures in several well-known non-projective dependency treebanks are binarizable without any increase in their fan-out.

More in general, it has been shown by Seki et al. (1991) that parsing of LCFRS can be carried out in time $\mathcal{O}(n^{|p_M|})$, where n is the length of the input string and p_M is the production in the grammar with largest size.[3] Thus, there may be cases in which one has to find an optimal tradeoff between rank and fan-out, in order to minimize the size of p_M. This requires some kind of Viterbi search over the space of all possible binarizations, constructed as described at the end of Subsection 4.3, for some appropriate value of the fan-out f'.

[3] The result has been shown for the formalism of multiple context-free grammars (MCFG), but it also applies to LCFRS, which are a special case of MCFG.

5.2 Extension to general LCFRS

This paper has focussed on *string-based* LCFRS. As discussed in Vijay-Shanker et al. (1987), LCFRS provide a more general framework where the productions are viewed as generating a set of abstract *derivation* trees. These trees can be used to specify how structures other than tuples of strings are composed. For example, LCFRS derivation trees can be used to specify how the elementary trees of a Tree Adjoining Grammar can be composed to produced derived tree. However, the results in this paper also apply to non-string-based LCFRS, since by limiting attention to the terminal string yield of whatever structures are under consideration, the composition operations can be defined using the string-based version of LCFRS that is discussed here.

5.3 Similar algorithmic techniques

The NAIVE-BINARIZATION algorithm given in Figure 1 is not novel to this paper: it is similar to an algorithm developed in Melamed et al. (2004) for generalized multitext grammars, a formalism weakly equivalent to LCFRS that has been introduced for syntax-based machine translation. However, the grammar produced by our algorithm has optimal (minimal) fan-out. This is an important improvement over the result in (Melamed et al., 2004), as this quantity enters into the parsing complexity of both multitext grammars and LCFRS as an exponential factor, and therefore must be kept as low as possible to ensure practically viable parsing.

Rank reduction is also investigated in Nesson et al. (2008) for synchronous tree-adjoining grammars, a synchronous rewriting formalism based on tree-adjoining grammars Joshi and Schabes (1992). In this case the search space of possible reductions is strongly restricted by the tree structures specified by the formalism, resulting in simplified computation for the reduction algorithms. This feature is not present in the case of LCFRS.

There is a close parallel between the technique used in the MINIMAL-BINARIZATION algorithm and deductive parsing techniques as proposed by Shieber et al. (1995), that are usually implemented by means of tabular methods. The idea of exploiting tabular parsing in production factorization was first expressed in Zhang et al. (2006). In fact, the particular approach presented here has been used to improve efficiency of parsing algorithms that use discontinuous syntactic models, in particular, non-projective dependency grammars, as discussed in Gómez-Rodríguez et al. (2009).

5.4 Open problems

The bounded binarization algorithm that we have presented has exponential run-time in the value of the input fan-out bound f'. It remains an open question whether the bounded binarization problem for LCFRS can be solved in deterministic polynomial time. Even in the restricted case of $f' = \varphi(p)$, that is, when no increase in the fan-out of the input production is allowed, we do not know whether p can be binarized using only deterministic polynomial time in the value of p's fan-out. However, our bounded binarization algorithm shows that the latter problem can be solved in polynomial time when the fan-out of the input LCFRS is bounded by some constant.

Whether the bounded binarization problem can be solved in polynomial time in the value of the input bound f' is also an open problem in the restricted case of synchronous context-free grammars, a special case of an LCFRS of fan-out two with a strict separation between the two components of each nonterminal in the right-hand side of a production, as discussed in the introduction. An interesting analysis of this restricted problem can be found in Gildea and Stefankovic (2007).

Acknowledgements The work of Carlos Gómez-Rodríguez was funded by Ministerio de Educación y Ciencia and FEDER (HUM2007-66607-C04) and Xunta de Galicia (PGIDIT07SIN005206PR, INCITE08E1R104022ES, INCITE08ENA305025ES, INCITE08PXIB302179PR and Rede Galega de Procesamento da Linguaxe e Recuperación de Información). The work of Marco Kuhlmann was funded by the Swedish Research Council. The work of Giorgio Satta was supported by MIUR under project PRIN No. 2007TJNZRE_002. We are grateful to an anonymous reviewer for a very detailed review with a number of particularly useful suggestions.

References

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Daniel Gildea and Daniel Stefankovic. 2007. Worst-case synchronous grammar rules. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 147–154. Association for Computational Linguistics, Rochester, New York.

Carlos Gómez-Rodríguez, David J. Weir, and John Carroll. 2009. Parsing mildly non-projective dependency structures. In *Twelfth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. To appear.

A. K. Joshi and Y. Schabes. 1992. Tree adjoining grammars and lexicalized grammars. In M. Nivat and A. Podelsky, editors, *Tree Automata and Languages*. Elsevier, Amsterdam, The Netherlands.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL), Main Conference Poster Sessions*, pages 507–514. Sydney, Australia.

Marco Kuhlmann and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Twelfth Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. To appear.

I. Dan Melamed, Benjamin Wellington, and Giorgio Satta. 2004. Generalized multitext grammars. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 661–668. Barcelona, Spain.

Rebecca Nesson, Giorgio Satta, and Stuart M. Shieber. 2008. Optimal k-arization of synchronous tree-adjoining grammar. In *Proceedings of ACL-08: HLT*, pages 604–612. Association for Computational Linguistics, Columbus, Ohio.

A. Nijholt. 1980. *Context-Free Grammars: Covers, Normal Forms, and Parsing*, volume 93. Springer-Verlag, Berlin, Germany.

Owen Rambow and Giorgio Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science*, 223(1–2):87–120.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On Multiple Context-Free Grammars. *Theoretical Computer Science*, 88(2):191–229.

Stuart M. Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36.

Takeaki Uno and Mutsunori Yagiura. 2000. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 104–111. Stanford, CA, USA.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, pages 977–984. Sydney, Australia.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting synchronous grammar rules from word-level alignments in linear time. In *22nd International Conference on Computational Linguistics (Coling)*, pages 1081–1088. Manchester, England, UK.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 256–263. New York, USA.

Inducing Compact but Accurate Tree-Substitution Grammars

Trevor Cohn and **Sharon Goldwater** and **Phil Blunsom**
School of Informatics
University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
Scotland, United Kingdom
{tcohn,sgwater,pblunsom}@inf.ed.ac.uk

Abstract

Tree substitution grammars (TSGs) are a compelling alternative to context-free grammars for modelling syntax. However, many popular techniques for estimating weighted TSGs (under the moniker of Data Oriented Parsing) suffer from the problems of inconsistency and over-fitting. We present a theoretically principled model which solves these problems using a Bayesian non-parametric formulation. Our model learns compact and simple grammars, uncovering latent linguistic structures (e.g., verb subcategorisation), and in doing so far out-performs a standard PCFG.

1 Introduction

Many successful models of syntax are based on *Probabilistic Context Free Grammars* (PCFGs) (e.g., Collins (1999)). However, directly learning a PCFG from a treebank results in poor parsing performance, due largely to the unrealistic independence assumptions imposed by the context-free assumption. Considerable effort is required to coax good results from a PCFG, in the form of grammar engineering, feature selection and clever smoothing (Collins, 1999; Charniak, 2000; Charniak and Johnson, 2005; Johnson, 1998). This effort must be repeated when moving to different languages, grammar formalisms or treebanks. We propose that much of this hand-coded knowledge can be obtained automatically as an emergent property of the treebanked data, thereby reducing the need for human input in crafting the grammar.

We present a model for automatically learning a *Probabilistic Tree Substitution Grammar* (PTSG), an extension to the PCFG in which non-terminals can rewrite as entire tree fragments (*elementary trees*), not just immediate children. These large fragments can be used to encode non-local context, such as head-lexicalisation and verb sub-categorisation. Since no annotated data is available providing TSG derivations we must induce the PTSG productions and their probabilities in an unsupervised way from an ordinary treebank. This is the same problem addressed by Data Oriented Parsing (DOP, Bod et al. (2003)), a method which uses as productions all subtrees of the training corpus. However, many of the DOP estimation methods have serious shortcomings (Johnson, 2002), namely inconsistency for DOP1 (Bod, 2003) and overfitting of the maximum likelihood estimate (Prescher et al., 2004).

In this paper we develop an alternative means of learning a PTSG from a treebanked corpus, with the twin objectives of a) finding a grammar which accurately models the data and b) keeping the grammar as simple as possible, with few, compact, elementary trees. This is achieved using a prior to encourage sparsity and simplicity in a Bayesian non-parametric formulation. The framework allows us to perform inference over an infinite space of grammar productions in an elegant and efficient manner. The net result is a grammar which only uses the increased context afforded by the TSG when necessary to model the data, and otherwise uses context-free rules.[1] That is, our model learns to use larger rules when the CFG's independence assumptions do not hold. This contrasts with DOP, which seeks to use all elementary trees from the training set. While our model is able, in theory, to use all such trees, in practice the data does not justify such a large grammar. Grammars that are only about twice the size of a

[1] While TSGs and CFGs describe the same string languages, TSGs can describe context-sensitive tree-languages, which CFGs cannot.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 548–556,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

treebank PCFG provide large gains in accuracy. We obtain additional improvements with grammars that are somewhat larger, but still much smaller than the DOP all-subtrees grammar. The rules in these grammars are intuitive, potentially offering insights into grammatical structure which could be used in, e.g., the development of syntactic ontologies and guidelines for future treebanking projects.

2 Background and related work

A *Tree Substitution Grammar*[2] (TSG) is a 4-tuple, $G = (T, N, S, R)$, where T is a set of *terminal symbols*, N is a set of *non-terminal symbols*, $S \in N$ is the distinguished *root non-terminal* and R is a set of productions (a.k.a. rules). The productions take the form of *elementary trees* – tree fragments of depth ≥ 2, where each internal node is labelled with a non-terminal and each leaf is labelled with either a terminal or a non-terminal. Non-terminal leaves are called *frontier non-terminals* and form the substitution sites in the generative process of creating trees with the grammar.

A *derivation* creates a tree by starting with the root symbol and rewriting (substituting) it with an elementary tree, then continuing to rewrite frontier non-terminals with elementary trees until there are no remaining frontier non-terminals. Unlike Context Free Grammars (CFGs) a syntax tree may not uniquely specify the derivation, as illustrated in Figure 1 which shows two derivations using different elementary trees to produce the same tree.

A *Probabilistic Tree Substitution Grammar* (PTSG), like a PCFG, assigns a probability to each rule in the grammar. The probability of a derivation is the product of the probabilities of its component rules, and the probability of a tree is the sum of the probabilities of its derivations.

As we mentioned in the introduction, work within the DOP framework seeks to induce PTSGs from treebanks by using all possible subtrees as rules, and one of a variety of methods for estimating rule probabilities.[3] Our aim of inducing compact grammars contrasts with that of DOP; moreover, we develop a probabilistic estimator which avoids the shortcomings of DOP1 and the maximum likelihood estimate (Bod, 2000; Bod, 2003; Johnson, 2002). Recent work on DOP estimation also seeks to address these problems, drawing from estimation theory to solve the consistency problem (Prescher et al., 2004; Zollmann and Sima'an, 2005), or incorporating a grammar brevity term into the learning objective (Zuidema, 2007). Our work differs from these previous approaches in that we explicitly model a prior over grammars within a Bayesian framework.[4]

Models of grammar refinement (Petrov et al., 2006; Liang et al., 2007; Finkel et al., 2007) also aim to automatically learn latent structure underlying treebanked data. These models allow each non-terminal to be split into a number of subcategories. Theoretically the grammar space of our model is a sub-space of theirs (projecting the TSG's elementary trees into CFG rules). However, the number of non-terminals required to recreate our TSG grammars in a PCFG would be exorbitant. Consequently, our model should be better able to learn specific lexical patterns, such as full noun-phrases and verbs with their sub-categorisation frames, while theirs are better suited to learning subcategories with larger membership, such as the terminals for days of the week and noun-adjective agreement. The approaches are orthogonal, and we expect that combining a category refinement model with our TSG model would provide better performance than either approach alone.

Our model is similar to the Adaptor Grammar model of Johnson et al. (2007b), which is also a kind of Bayesian nonparametric tree-substitution grammar. However, Adaptor Grammars require that each sub-tree expands completely, with only terminal symbols as leaves, while our own model permits non-terminal frontier nodes. In addition, they disallow recursive containment of adapted non-terminals; we impose no such constraint.

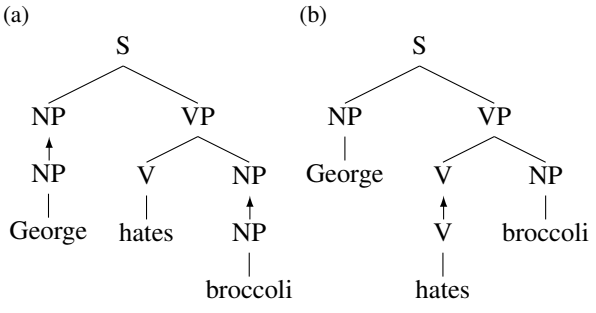
3 Model

Recall the nature of our task: we are given a corpus of parse trees $\mathbf{t}$ and wish to infer a tree-substitution grammar G that we can use to parse new data. Rather than inferring a grammar directly, we go through an intermediate step of inferring a distribution over the derivations used to produce $\mathbf{t}$, i.e.,

[2] A TSG is a *Tree Adjoining Grammar* (TAG; Joshi (2003)) without the adjunction operator.

[3] TAG induction (Chiang and Bikel, 2002; Xia, 2002) also tackles a similar learning problem.

[4] A similar Bayesian model of TSG induction has been developed independently to this work (O'Donnell et al., 2009b; O'Donnell et al., 2009a).



S → NP (VP (V hates) NP)
NP → George
NP → broccoli
S → (NP George) (VP V (NP broccoli))
V → hates

Figure 1: Example derivations for the same tree, where arrows indicate substitution sites. The elementary trees used in (a) and (b) are shown below as grammar productions in bracketed tree notation.

a distribution over sequences of elementary trees $\mathbf{e}$ that compose to form $\mathbf{t}$. We will then essentially read the grammar off the elementary trees, as described in Section 5. Our problem therefore becomes one of identifying the posterior distribution of $\mathbf{e}$ given $\mathbf{t}$, which we can do using Bayes' Rule:

$$P(\mathbf{e}|\mathbf{t}) \propto P(\mathbf{t}|\mathbf{e})P(\mathbf{e}) \quad (1)$$

Since the sequence of elementary trees can be split into derivations, each of which completely specifies a tree, $P(\mathbf{t}|\mathbf{e})$ is either equal to 1 (when $\mathbf{t}$ and $\mathbf{e}$ are consistent) or 0 (otherwise). Therefore, the work in our model is done by the prior distribution over elementary trees. Note that this is analogous to the Bayesian model of word segmentation presented by Goldwater et al. (2006); indeed, the problem of inferring $\mathbf{e}$ from $\mathbf{t}$ can be viewed as a segmentation problem, where each full tree must be segmented into one or more elementary trees. As in Goldwater et al. (2006), we wish to favour solutions employing a relatively small number of elementary units (here, elementary trees). This can be done using a Dirichlet process (DP) prior. Specifically, we define the distribution of elementary tree e with root non-terminal symbol c as

$$\begin{aligned} G_c|\alpha_c, P_0 &\sim \text{DP}(\alpha_c, P_0(\cdot|c)) \\ e|c &\sim G_c \end{aligned}$$

where $P_0(\cdot|c)$ (the *base distribution*) is a distribution over the infinite space of trees rooted with c, and α_c (the *concentration parameter*) controls the model's tendency towards either reusing elementary trees or creating novel ones as each training instance is encountered (and consequently, the tendency to infer larger or smaller sets of elementary trees from the observed data). We discuss the base distribution in more detail below.

Rather than representing the distribution G_c explicitly, we integrate over all possible values of G_c. The resulting distribution over e_i, conditioned on $\mathbf{e}_{<i} = e_1 \ldots e_{i-1}$ and the root category c is:

$$p(e_i|\mathbf{e}_{<i}, c, \alpha_c, P_0) = \frac{n^{<i}_{e_i,c} + \alpha_c P_0(e_i|c)}{n^{<i}_{\cdot,c} + \alpha_c} \quad (2)$$

where $n^{<i}_{e_i,c}$ is the number number of times e_i has been used to rewrite c in $\mathbf{e}_{<i}$, and $n^{<i}_{\cdot,c} = \sum_e n^{<i}_{e,c}$ is the total count of rewriting c.

As with other DP models, ours can be viewed as a *cache model*, where e_i can be generated in one of two ways: by drawing from the base distribution, where the probability of any particular tree is proportional to $\alpha_c P_0(e_i|c)$, or by drawing from a cache of previous expansions of c, where the probability of any particular expansion is proportional to the number of times that expansion has been used before. This view makes it clear that the model embodies a "rich-get-richer" dynamic in which a few expansions will occur with high probability, but many will occur only once or twice, as is typical of natural language. Our model is similar in this way to the Adaptor Grammar model of Johnson et al. (2007a).

We still need to define P_0, the base distribution over tree fragments. We use two such distributions. The first, P_0^M generates each elementary tree by a series of random decisions: whether to expand a non-terminal, how many children to produce and their identities. The probability of expanding a non-terminal node labelled c is parameterised via a binomial distribution, $\text{Bin}(\beta_c)$, while all other decisions are chosen uniformly at random. The second base distribution, P_0^C, has a similar generative process but draws non-terminal expansions from a treebank-trained PCFG instead of a uniform distribution.

Both choices of P_0 have the effect of biasing the model towards simple rules with a small number of internal nodes. The geometric increase in cost discourages the model from using larger rules; for this to occur these rules must yield a large increase in the data likelihood. As P_0^C incorporates PCFG probabil-

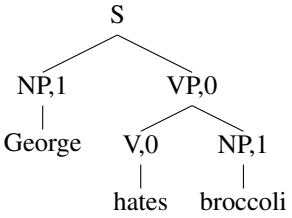


Figure 2: Gibbs state e specifying the derivation in Figure 1a. Each node is labelled with its substitution indicator variable.

ities, it assigns higher relative probability to larger rules, compared to the more draconian P_0^M.

4 Training

To train our model we use Gibbs sampling (Geman and Geman, 1984), a Markov chain Monte Carlo method in which variables are repeatedly sampled conditioned on the values of all other variables in the model. After a period of burn-in, each sampler state (set of variable assignments) is a sample from the posterior distribution of the model. In our case, we wish to sample from $P(\mathbf{e}|\mathbf{t}, \alpha, \beta)$, where $(\alpha, \beta) = \{\alpha_c, \beta_c\}$ for all categories c. To do so, we associate a binary variable with each non-root internal node of each tree in the training set, indicating whether that node is a substitution point or not. Each substitution point forms the root of some elementary tree, as well as a frontier non-terminal of an ancestor node's elementary tree. Collectively, the training trees and substitution variables specify the sequence of elementary trees $\mathbf{e}$ that is the current state of the sampler. Figure 2 shows an example tree with its substitution variables, corresponding to the TSG derivation in Figure 1a.

Our Gibbs sampler works by sampling the value of each substitution variable, one at a time, in random order. If d is the node associated with the substitution variable s under consideration, then the two possible values of s define two options for $\mathbf{e}$: one in which d is internal to some elementary tree e_M, and one in which d is the substitution site connecting two smaller trees, e_A and e_B. In the example in Figure 2, when sampling the VP node, e_M = (S NP (VP (V hates) NP)), e_A = (S NP VP), and e_B = (VP (V hates) NP). To sample a value for s, we compute the probabilities of e_M and (e_A, e_B), conditioned on $\mathbf{e}^-$: all other elementary trees in the training set that share at most a root or frontier non-terminal with e_M, e_A, or e_B. This is easy to do because the DP is *exchangeable*, meaning that the probability of a set of outcomes does not depend on their ordering. Therefore, we can treat the elementary trees under consideration as the last ones to be sampled, and apply Equation 2, giving us

$$P(e_M|c_M) = \frac{n^-_{e_M,c_M} + \alpha_{c_M} P_0(e_M|c_M)}{n^-_{\cdot,c_M} + \alpha_{c_M}} \quad (3)$$

$$P(e_A, e_B|c_A) = \frac{n^-_{e_A,c_A} + \alpha_{c_A} P_0(e_A|c_A)}{n^-_{\cdot,c_A} + \alpha_{c_A}} \times \frac{n^-_{e_B,c_B} + \delta(e_A, e_B) + \alpha_{c_B} P_0(e_B|c_B)}{n^-_{\cdot,c_B} + \delta(c_A, c_B) + \alpha_{c_B}} \quad (4)$$

where c_x is the root label of $e_x, x \in \{A, B, M\}$, the counts n^- are with respect to $\mathbf{e}^-$, and $\delta(\cdot,\cdot)$ is the Kronecker delta function, which returns 1 when its arguments are identical and 0 otherwise. We have omitted $\mathbf{e}^-, \mathbf{t}, \alpha$ and β from the conditioning context. The δ terms in the second factor of (4) account the changes to n^- that would occur after observing e_A, which forms part of the conditioning context for e_B. If the trees e_A and e_B are identical, then the count $n^-_{e_B,c_B}$ would increase by one, and if the trees share the same root non-terminal, then $n^-_{\cdot,c_B}$ would increase by one.

In the previous discussion, we have assumed that the model hyperparameters, (α, β), are known. However, selecting their values by hand is extremely difficult and fitting their values on heldout data is often very time consuming. For this reason we treat the hyper-parameters as variables in our model and infer their values during training. We choose vague priors for each hyper-parameter, encoding our lack of information about their values. We treat the concentration parameters, α, as being generated by a vague gamma prior, $\alpha_c \sim \text{Gamma}(0.001, 1000)$. We sample a new value α'_c using a log-normal distribution with mean α_c and variance 0.3, which is then accepted into the distribution $p(\alpha_c|\mathbf{e}, \mathbf{t}, \alpha^-, \beta)$ using the Metropolis-Hastings algorithm. We use a Beta prior for the binomial specification parameters, $\beta_c \sim \text{Beta}(1, 1)$. As the Beta distribution is conjugate to the binomial, we can directly resample the β parameters from the posterior, $p(\beta_c|\mathbf{e}, \mathbf{t}, \alpha, \beta^-)$. Both the concentration and substitution parameters are resampled after every full Gibbs sampling iteration over the training trees.

5 Parsing

We now turn to the problem of using the model to parse novel sentences. This requires finding the maximiser of

$$p(t|w,\mathbf{t}) = \int p(t|w,\mathbf{e},\alpha,\beta)\; p(\mathbf{e},\alpha,\beta|\mathbf{t})\; d\mathbf{e}\; d\alpha\; d\beta \tag{5}$$

where w is the sequence of words being parsed and t the resulting tree, $\mathbf{t}$ are the training trees and $\mathbf{e}$ their segmentation into elementary trees.

Unfortunately solving for the maximising parse tree in (5) is intractable. However, it can approximated using Monte Carlo techniques. Given a sample of $(\mathbf{e},\alpha,\beta)$[5] we can reason over the space of possible trees using a Metropolis-Hastings sampler (Johnson et al., 2007a) coupled with a Monte Carlo integral (Bod, 2003). The first step is to sample from the posterior over derivations, $p(d|w,\mathbf{e},\alpha,\beta)$. This is achieved by drawing samples from an approximation grammar, $\tilde{p}(d|w)$, which are then accepted to the true distribution using the Metropolis-Hastings algorithm. The second step records for each sampled derivation the CFG tree. The counts of trees constitute an approximation to $p(t|w,\mathbf{e},\alpha,\beta)$, from which we can recover the maximum probability tree.

A natural proposal distribution, $\tilde{p}(d|w)$, is the maximum a posterior (MAP) grammar given the elementary tree analysis of our training set (analogous to the PCFG approximation used in Johnson et al. (2007a)). This is not practical because the approximation grammar is infinite: elementary trees with zero count in $\mathbf{e}$ still have some residual probability under P_0. In the absence of a better alternative, we discard (most of) the zero-count rules from MAP grammar. This results in a tractable grammar representing the majority of the probability mass, from which we can sample derivations. We specifically retain all zero-count PCFG productions observed in the training set in order to provide greater robustness on unseen data.

In addition to finding the maximum probability parse (MPP), we also report results using the maximum probability derivation (MPD). While this could be calculated in the manner as described above, we found that using the CYK algorithm (Cocke, 1969) to find the Viterbi derivation for $\tilde{p}$ yielded consistently better results. This algorithm maximises an approximated model, as opposed to approximately optimising the true model. We also present results using the tree with the maximum expected count of CFG rules (MER). This uses counts of the CFG rules applied at each span (compiled from the derivation samples) followed by a maximisation step to find the best tree. This is similar to the MAX-RULE-SUM algorithm of Petrov and Klein (2007) and maximum expected recall parsing (Goodman, 2003).

[5]Using many samples of $(\mathbf{e},\alpha,\beta)$ in a Monte Carlo integral is a straight-forward extension to our parsing algorithm. We did not observe a significant improvement in parsing accuracy when using a multiple samples compared to a single sample, and therefore just present results for a single sample.

$$S \rightarrow A \mid B$$
$$A \rightarrow A\ A \mid B\ B \mid (A\ a)\ (A\ a) \mid (B\ a)\ (B\ a)$$
$$B \rightarrow A\ A \mid B\ B \mid (A\ b)\ (A\ b) \mid (B\ b)\ (B\ b)$$

Figure 3: TSG used to generate synthetic data. All production probabilities are uniform.

6 Experiments

Synthetic data Before applying the model to natural language, we first create a synthetic problem to confirm that the model is capable of recovering a known tree-substitution grammar. We created 50 random trees from the TSG shown in Figure 3. This produces binary trees with A and B internal nodes and 'a' and 'b' as terminals, such that the terminals correspond to their grand-parent non-terminal (A and a or B and b). These trees cannot be modelled accurately with a CFG because expanding A and B nodes into terminal strings requires knowing their parent's non-terminal.

We train the model for 100 iterations of Gibbs sampling using annealing to speed convergence. Annealing amounts to smoothing the distributions in (3) and (4) by raising them to the power of $\frac{1}{T}$. Our annealing schedule begins at $T = 3$ and linearly decreases to reach $T = 1$ in the final iteration. The sampler converges to the correct grammar, with the 10 rules from Figure 3.

Penn-treebank parsing We ran our natural language experiments on the Penn treebank, using the standard data splits (sections 2–21 for training, 22 for development and 23 for testing). As our model is parameter free (the α and β parameters are learnt in training), we do not use the development set for pa-

rameter tuning. We expect that fitting these parameters to maximise performance on the development set would lead to a small increase in generalisation performance, but at a significant cost in runtime. We replace tokens with count ≤ 1 in the training sample with one of roughly 50 generic unknown word markers which convey the token's lexical features and position in the sentence, following Petrov et al. (2006). We also right-binarise the trees to reduce the branching factor in the same manner as Petrov et al. (2006). The predicted trees are evaluated using EVALB[6] and we report the F1 score over labelled constituents and exact match accuracy over all sentences in the testing sets.

In our experiments, we initialised the sampler by setting all substitution variables to 0, thus treating every full tree in the training set as an elementary tree. Starting with all the variables set to 1 (corresponding to CFG expansions) or a random mix of 0s and 1s considerably increases time until convergence. We hypothesise that this is due to the sampler getting stuck in modes, from which a series of locally bad decisions are required to escape. The CFG solution seems to be a mode and therefore starting the sampler with maximal trees helps the model to avoid this mode.

Small data sample For our first treebank experiments, we train on a small data sample by using only section 2 of the treebank. Bayesian methods tend to do well with small data samples, while for larger samples the benefits diminish relative to point estimates. The models were trained using Gibbs sampling for 4000 iterations with annealing linearly decreasing from $T = 5$ to $T = 1$, after which the model performed another 1000 iterations with $T = 1$. The final training sample was used in the parsing algorithm, which used 1000 derivation samples for each test sentence. All results are the average of five independent runs.

Table 1 presents the prediction results on the development set. The baseline is a maximum likelihood PCFG. The TSG model significantly outperforms the baseline with either base distribution P_0^M or P_0^C. This confirms our hypothesis that CFGs are not sufficiently powerful to model syntax, but that the increased context afforded to the TSG can make a large difference. This result is even more impressive when considering the difference in the sizes of grammar in the PCFG versus TSG models. The TSG using P_0^M achieves its improvements with only double as many rules, as a consequence of the prior which encourages sparse solutions. The TSG results with the CFG base distribution, P_0^C, are more accurate but with larger grammars.[7] This base distribution assigns proportionally higher probability to larger rules than P_0^M, and consequently the model uses these additional rules in a larger grammar.

		F1	EX	# rules
PCFG		60.20	4.29	3500
TSG P_0^M:	MPD	72.17	11.92	6609
	MPP	71.27	12.33	6609
	MER	74.25	12.30	6609
TSG P_0^C:	MPD	75.24	15.18	14923
	MPP	75.30	15.74	14923
	MER	76.89	15.76	14923
$SM_{\tau=2}$:	MPD	71.93	11.30	16168
	MER	74.32	11.77	16168
$SM_{\tau=5}$:	MPD	75.33	15.64	39758
	MER	77.93	16.94	39758

Table 1: Development results for models trained on section 2 of the Penn tree-bank, showing labelled constituent F1 and exact match accuracy. Grammar sizes are the number of rules with count ≥ 1.

Surprisingly, the MPP technique is not systematically better than the MPD approach, with mixed results under the F1 metric. We conjecture that this is due to sampling variance for long sentences, where repeated samples of the same tree are exceedingly rare. The MER technique results in considerably better F1 scores than either MPD or MPP, with a margin of 1.5 to 3 points. This method is less affected by sampling variance due to its use of smaller tree fragments (PCFG productions at each span).

For comparison, we trained the Berkeley split-merge (SM) parser (Petrov et al., 2006) on the same data and decoded using the Viterbi algorithm (MPD) and expected rule count (MER a.k.a. MAX-RULE-SUM). We ran two iterations of split-merge training, after which the development F1 dropped substantially (in contrast, our model is not fit to the development data). The result is an accuracy slightly below that of our model ($SM_{\tau=2}$). To be fairer to their model, we adjusted the unknown word threshold to their default setting, i.e., to apply to word types oc-

[6] See `http://nlp.cs.nyu.edu/evalb/`.

[7] The grammar is nevertheless far smaller than the full DOP grammar on this data set, which has 700K rules.

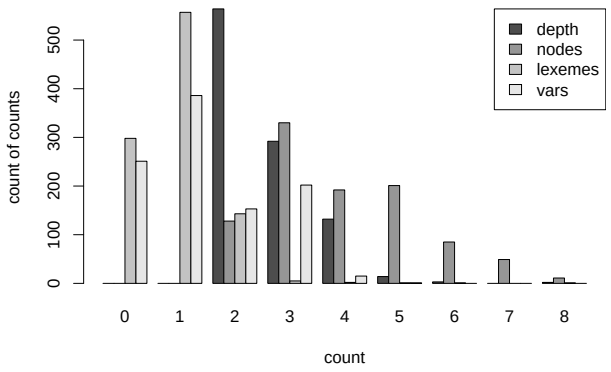


Figure 4: Grammar statistics for a TSG P_0^M model trained on section 2 of the Penn treebank, showing a histogram over elementary tree depth, number of nodes, terminals (lexemes) and frontier nonterminals (vars).

curring fewer than five times ($SM_{\tau=5}$). We expect that tuning the treatment of unknown words in our model would also yield further gains. The grammar sizes are not strictly comparable, as the Berkeley binarised grammars prohibit non-binary rules, and are therefore forced to decompose each of these rules into many child rules. But the trend is clear – our model produces similar results to a state-of-the-art parser, and can do so using a small grammar. With additional rounds of split-merge training, the Berkeley grammar grows exponentially larger (200K rules after six iterations).

Full treebank We now train the model using P_0^M on the full training partition of the Penn treebank, using sections 2–21. We run the Gibbs sampler for 15,000 iterations while annealing from $T = 5$ to $T = 1$, after which we finish with 5,000 iterations at $T = 1$. We repeat this three times, giving an average F1 of 84.0% on the testing partition using the maximum expected rule algorithm and 83.0% using the Viterbi algorithm. This far surpasses the ML-PCFG (F1 of 70.7%), and is similar to Zuidema's (2007) DOP result of 83.8%. However, it still well below state-of-the art parsers (e.g., the Berkeley parser trained using the same data representation scores 87.7%). But we must bear in mind that these parsers have benefited from years of tuning to the Penn-treebank, where our model is much simpler and is largely untuned. We anticipate that careful data preparation and model tuning could greatly improve our model's performance.

NP →
(NNP Mr.) NNP
CD (NN %)
(NP CD (NN %)) (PP (IN of) NP)
(NP ($ $) CD) (NP (DT a) (NN share))
(NP (DT the) (N̄P (NN company) POS)) N̄P
(NP QP (NN %)) (PP (IN of) NP)
(NP CD (NNS cents)) (NP (DT a) (NN share))
(NP (NNP Mr.) (N̄P NNP (POS 's))) NN
QP (NN %)
(NP (NN president)) (PP (IN of) NP)
(NP (NNP Mr.) (N̄P NNP (POS 's))) N̄P
NNP (N̄P NNP (NNP Corp.))
NNP (N̄P NNP (NNP Inc.))
(NP (NN chairman)) (PP (IN of) NP)

VP →
(VBD said) (SBAR (S (NP (PRP it)) VP))
(VBD said) (SBAR (S NP VP))
(VBD rose) (V̄P (NP CD (NN %)) V̄P)
(VBP want) S
(VBD said) (SBAR (S (NP (PRP he)) VP))
(VBZ plans) S
(VBD said) (SBAR S)
(VBZ says) (SBAR (S NP VP))
(VBP think) (SBAR S)
(VBD agreed) (S (VP (TO to) (VP VB V̄P)))
(VBZ includes) NP
(VBZ says) (SBAR (S (NP (PRP he)) VP))
(VBZ wants) S
(VBD closed) (V̄P (PP (IN at) NP) (V̄P , ADVP))

Table 3: Most frequent lexicalised expansions for noun and verb phrases, excluding auxiliary verbs.

7 Discussion

So what kinds of non-CFG rules is the model learning? Figure 4 shows the grammar statistics for a TSG model trained on the small data sample. This model has 5611 CFG rules and 1008 TSG rules. The TSG rules vary in depth from two to nine levels with the majority between two and four. Most rules combine a small degree of lexicalisation and a variable or two. This confirms that the model is learning local structures to encode, e.g., multi-word units, subcategorisation frames and lexical agreement. The few very large rules specify full parses for sentences which were repeated in the training corpus. These complete trees are also evident in the long tail of node counts (up to 27; not shown in the figure) and counts for highly lexicalised rules (up to 8).

To get a better feel for the types of rules being learnt, it is instructive to examine the rules in the re-

NP →	PP →	ADJP →
DT $\bar{\text{NP}}$	IN NP	JJ
NNS	(IN in) NP	RB JJ
DT NN	(TO to) NP	JJ ($\bar{\text{ADJP}}$ CC JJ)
(DT the) $\bar{\text{NP}}$	TO NP	JJ PP
JJ NNS	(IN with) NP	(RB very) JJ
NP (PP (IN of) NP)	(IN of) NP	RB $\bar{\text{ADJP}}$
NP PP	(IN by) NP	(RBR more) JJ
NP ($\bar{\text{NP}}$ (CC and) NP)	(IN at) NP	JJ $\bar{\text{ADJP}}$
JJ $\bar{\text{NP}}$	IN (NP (DT the) $\bar{\text{NP}}$)	ADJP ($\bar{\text{ADJP}}$ CC ADJP)
NN NNS	(IN on) NP	RB VBN
(DT the) NNS	(IN from) NP	RB ($\bar{\text{ADJP}}$ JJ PP)
DT ($\bar{\text{NP}}$ JJ NN)	IN (S (VP VBG NP))	JJ (PP (TO to) NP)
NN	IN (NP NP PP)	ADJP (PP (IN than) NP)
JJ NN	(IN into) NP	(RB too) JJ
(NP DT NN) (PP (IN of) NP)	(IN for) NP	(RB much) JJR

Table 2: Top fifteen expansions sorted by frequency (most frequent at top), taken from the final sample of a model trained on the full Penn treebank. Non-terminals shown with an over-bar denote a binarised sub span of the given phrase type.

sultant grammar. Table 2 shows the top fifteen rules for three phrasal categories for the model trained on the full Penn treebank. We can see that many of these rules are larger than CFG rules, showing that the CFG rules alone are inadequate to model the treebank. Two of the NP rules encode the prevalence of preposition phrases headed by 'of' within a noun phrase, as opposed to other prepositions. Also noteworthy is the lexicalisation of the determiner, which can affect the type of NP expansion. For instance, the indefinite article is more likely to have an adjectival modifier, while the definite article appears more frequently unmodified. Highly specific tokens are also incorporated into lexicalised rules.

Many of the verb phrase expansions have been lexicalised, encoding the verb's subcategorisation, as shown in Table 3. Notice that each verb here accepts only one or a small set of argument frames, indicating that by lexicalising the verb in the VP expansion the model can find a less ambiguous and more parsimonious grammar.

The model also learns to use large rules to describe the majority of root node expansions (we add a distinguished TOP node to all trees). These rules mostly describe cases when the S category is used for a full sentence, which most often include punctuation such as the full stop and quotation marks. In contrast, the majority of expansions for the S category do not include any punctuation. The model has learnt to differentiate between the two different classes of S – full sentence versus internal clause – due to their different expansions.

8 Conclusion

In this work we have presented a non-parametric Bayesian model for inducing tree substitution grammars. By incorporating a structured prior over elementary rules our model is able to reason over the infinite space of all such rules, producing compact and simple grammars. In doing so our model learns local structures for latent linguistic phenomena, such as verb subcategorisation and lexical agreement. Our experimental results show that the induced grammars strongly out-perform standard PCFGs, and are comparable to a state-of-the-art parser on small data samples. While our results on the full treebank are well shy of the best available parsers, we have proposed a number of improvements to the model and the parsing algorithm that could lead to state-of-the-art performance in the future.

References

Rens Bod, Remko Scha, and Khalil Sima'an, editors. 2003. *Data-oriented parsing*. Center for the Study of Language and Information - Studies in Computational Linguistics. University of Chicago Press.

Rens Bod. 2000. Combining semantic and syntactic structure for language modeling. In *Proceedings of the 6th International Conference on Spoken Language Processing*, Beijing, China.

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, Budapest, Hungary, April.

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*, pages 173–180, Ann Arbor, Michigan, June.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 183–189, Taipei, Taiwan.

John Cocke. 1969. *Programming languages and their compilers: Preliminary notes*. Courant Institute of Mathematical Sciences, New York University.

Michael John Collins. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2007. The infinite tree. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 272–279, Prague, Czech Republic, June.

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

Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of COLING/ACL*, Sydney.

Joshua Goodman. 2003. Efficient parsing of DOP with PCFG-reductions. In Bod et al. (Bod et al., 2003), chapter 8.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007a. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 139–146, Rochester, New York, April.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007b. Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems 19*.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4), December.

Mark Johnson. 2002. The DOP estimation method is biased and inconsistent. *Computational Lingusitics*, 28(1):71–76, March.

Aravind Joshi. 2003. Tree adjoining grammars. In Ruslan Mikkov, editor, *The Oxford Handbook of Computational Linguistics*, pages 483–501. Oxford University Press, Oxford, England.

Percy Liang, Slav Petrov, Michael Jordan, and Dan Klein. 2007. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697, Prague, Czech Republic, June.

Timothy J. O'Donnell, Noah D. Goodman, Jesse Snedeker, and Joshua B. Tenenbaum. 2009a. Computation and reuse in language. In *31st Annual Conference of the Cognitive Science Society*, Amsterdam, The Netherlands, July. To appear.

Timothy J. O'Donnell, Noah D. Goodman, and Joshua B. Tenenbaum. 2009b. Fragment grammar: Exploring reuse in hierarchical generative processes. Technical Report MIT-CSAIL-TR-2009-013, MIT.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July.

Detlef Prescher, Remko Scha, Khalil Sima'an, and Andreas Zollmann. 2004. On the statistical consistency of dop estimators. In *Proceedings of the 14th Meeting of Computational Linguistics in the Netherlands*, Antwerp, Belgium.

Fei Xia. 2002. *Automatic grammar generation from two different perspectives*. Ph.D. thesis, University of Pennsylvania.

Andreas Zollmann and Khalil Sima'an. 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, 10(2):367–388.

Willem Zuidema. 2007. Parsimonious data-oriented parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 551–560, Prague, Czech Republic, June.

Hierarchical Search for Parsing

Adam Pauls **Dan Klein**
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720, USA
{adpauls,klein}@cs.berkeley.edu

Abstract

Both coarse-to-fine and A* parsing use simple grammars to guide search in complex ones. We compare the two approaches in a common, agenda-based framework, demonstrating the tradeoffs and relative strengths of each method. Overall, coarse-to-fine is much faster for moderate levels of search errors, but below a certain threshold A* is superior. In addition, we present the first experiments on hierarchical A* parsing, in which computation of heuristics is itself guided by meta-heuristics. Multi-level hierarchies are helpful in both approaches, but are more effective in the coarse-to-fine case because of accumulated slack in A* heuristics.

1 Introduction

The grammars used by modern parsers are extremely large, rendering exhaustive parsing impractical. For example, the lexicalized grammars of Collins (1997) and Charniak (1997) and the state-split grammars of Petrov *et al.* (2006) are all too large to construct unpruned charts in memory. One effective approach is *coarse-to-fine* pruning, in which a small, coarse grammar is used to prune edges in a large, refined grammar (Charniak *et al.*, 2006). Indeed, coarse-to-fine is even more effective when a hierarchy of successive approximations is used (Charniak *et al.*, 2006; Petrov and Klein, 2007). In particular, Petrov and Klein (2007) generate a sequence of approximations to a highly subcategorized grammar, parsing with each in turn.

Despite its practical success, coarse-to-fine pruning is approximate, with no theoretical guarantees on optimality. Another line of work has explored *A* search* methods, in which simpler problems are used not for pruning, but for prioritizing work in the full search space (Klein and Manning, 2003a; Haghighi *et al.*, 2007). In particular, Klein and Manning (2003a) investigated A* for lexicalized parsing in a factored model. In that case, A* vastly improved the search in the lexicalized grammar, with provable optimality. However, their bottleneck was clearly shown to be the exhaustive parsing used to compute the A* heuristic itself. It is not obvious, however, how A* can be stacked in a hierarchical or multi-pass way to speed up the computation of such complex heuristics.

In this paper, we address three open questions regarding efficient hierarchical search. First, can a hierarchy of A* bounds be used, analogously to hierarchical coarse-to-fine pruning? We show that recent work in hierarchical A* (Felzenszwalb and McAllester, 2007) can naturally be applied to both the hierarchically refined grammars of Petrov and Klein (2007) as well as the lexicalized grammars of Klein and Manning (2003a). Second, what are the tradeoffs between coarse-to-fine pruning and A* methods? We show that coarse-to-fine is generally much faster, but at the cost of search errors.[1] Below a certain search error rate, A* is faster and, of course, optimal. Finally, when and how, qualitatively, do these methods fail? A* search's work grows quickly as the slack increases between the heuristic bounds and the true costs. On the other hand, coarse-to-fine prunes unreliably when the approximating grammar

[1] In this paper, we consider only errors made by the search procedure, not modeling errors.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 557–565,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

Name	Rule			Priority
IN	$r : w_r \;\; I(\mathrm{B}_t, i, k) : \beta_B \;\; I(\mathrm{C}_t, k, j) : \beta_C$	$\Rightarrow$	$I(\mathrm{A}_t, i, j) : \beta_A = \beta_B + \beta_C + w_r$	$\beta_A + h(A, i, j)$

Table 1: Deduction rule for A* parsing. The items on the left of the $\Rightarrow$ indicate what edges must be present on the chart and what rule can be used to combine them, and the item on the right is the edge that may be added to the agenda. The weight of each edge appears after the colon. The rule r is A $\rightarrow$ B C.

is very different from the target grammar. We empirically demonstrate both failure modes.

2 Parsing algorithms

Our primary goal in this paper is to compare hierarchical A* (HA*) and hierarchical coarse-to-fine (CTF) pruning methods. Unfortunately, these two algorithms are generally deployed in different architectures: CTF is most naturally implemented using a dynamic program like CKY, while best-first algorithms like A* are necessarily implemented with agenda-based parsers. To facilitate comparison, we would like to implement them in a common architecture. We therefore work entirely in an agenda-based setting, noting that the crucial property of CTF is not the CKY order of exploration, but the pruning of unlikely edges, which can be equally well done in an agenda-based parser. In fact, it is possible to closely mimic dynamic programs like CKY using a best-first algorithm with a particular choice of priorities; we discuss this in Section 2.3.

While a general HA* framework is presented in Felzenszwalb and McAllester (2007), we present here a specialization to the parsing problem. We first review the standard agenda-driven search framework and basic A* parsing before generalizing to HA*.

2.1 Agenda-Driven Parsing

A non-hierarchical, best-first parser takes as input a PCFG $\mathcal{G}$ (with root symbol R), a priority function $p(\cdot)$ and a sentence consisting of terminals (words) $\mathrm{T}^0 \ldots \mathrm{T}^{n-1}$. The parser's task is to find the best scoring (Viterbi) tree structure which is rooted at R and spans the input sentence. Without loss of generality, we consider grammars in Chomsky normal form, so that each non-terminal rule in the grammar has the form $r = \mathrm{A} \rightarrow \mathrm{B}\ \mathrm{C}$ with weight w_r. We assume that weights are non-negative (e.g. negative log probabilities) and that we wish to minimize the sum of the rule weights.

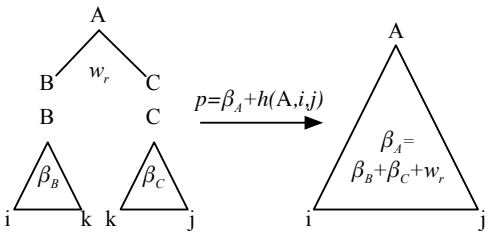


Figure 1: Deduction rule for A* depicted graphically. Items to the left of the arrow indicate edges and rules that can be combined to produce the edge to the right of the arrow. Edges are depicted as complete triangles. The value inside an edge represents the weight of that edge. Each new edge is assigned the priority written above the arrow when added to the agenda.

The objects in an agenda-based parser are *edges* $e = I(\mathrm{X}, i, j)$, also called *items*, which represent parses spanning i to j and rooted at symbol X. We denote edges as triangles, as in Figure 1. At all times, edges have *scores* β_e, which are estimates of their Viterbi inside probabilities (also called *path costs*). These estimates improve over time as new derivations are considered, and may or may not be correct at termination, depending on the properties of p. The parser maintains an *agenda* (a priority queue of edges), as well as a *chart* (or *closed list* in search terminology) of edges already processed. The fundamental operation of the algorithm is to pop the best (lowest) priority edge e from the agenda, put it into the chart, and enqueue any edges which can be built by combining e with other edges in the chart. The combination of two adjacent edges into a larger edge is shown graphically in Figure 1 and as a weighted deduction rule in Table 1 (Shieber *et al.*, 1995; Nederhof, 2003). When an edge a is built from adjacent edges b and c and a rule r, its current score β_a is compared to $\beta_b + \beta_c + w_r$ and updated if necessary. To allow reconstruction of best parses, backpointers are maintained in the standard way. The agenda is initialized with $I(\mathrm{T}^i, i, i+1)$

for $i = 0 \dots n-1$. The algorithm terminates when $I(\mathrm{R}, 0, n)$ is popped off the queue.

Priorities are in general different than weights. Whenever an edge e's score changes, its priority $p(e)$, which may or may not depend on its score, may improve. Edges are promoted accordingly in the agenda if their priorities improve. In the simplest case, the priorities are simply the β_e estimates, which gives a correct uniform cost search wherein the root edge is guaranteed to have its correct inside score estimate at termination (Caraballo and Charniak, 1996).

A* parsing (Klein and Manning, 2003b) is a special case of such an agenda-driven parser in which the priority function p takes the form $p(e) = \beta_e + h(e)$, where $e = I(\mathrm{X}, i, j)$ and $h(\cdot)$ is some approximation of e's Viterbi outside cost (its *completion cost*). If h is *consistent*, then the A* algorithm guarantees that whenever an edge comes off the agenda, its weight is its true Viterbi inside cost. In particular, this guarantee implies that the first edge representing the root $I(\mathrm{R}, 0, n)$ will be scored with the true Viterbi score for the sentence.

2.2 Hierarchical A*

In the standard A* case the heuristics are assumed to come from a black box. For example, Klein and Manning (2003b) precomputes most heuristics offline, while Klein and Manning (2003a) solves simpler parsing problems for each sentence. In such cases, the time spent to compute heuristics is often non-trivial. Indeed, it is typical that effective heuristics are themselves expensive search problems. We would therefore like to apply A* methods to the computation of the heuristics themselves. Hierarchical A* allows us to do exactly that.

Formally, HA* takes as input a sentence and a sequence (or *hierarchy*) of $m+1$ PCFGs $\mathcal{G}_0 \dots \mathcal{G}_m$, where $\mathcal{G}_m$ is the *target grammar* and $\mathcal{G}_0 \dots \mathcal{G}_{m-1}$ are *auxiliary grammars*. Each grammar $\mathcal{G}_t$ has an inventory of symbols Σ_t, hereafter denoted with capital letters. In particular, each grammar has a distinguished terminal symbol T_t^i for each word T^i in the input and a root symbol R_t.

The grammars $\mathcal{G}_0 \dots \mathcal{G}_m$ must form a hierarchy in which $\mathcal{G}_t$ is a *relaxed projection* of $\mathcal{G}_{t+1}$. A grammar $\mathcal{G}_{t-1}$ is a *projection* of $\mathcal{G}_t$ if there exists some onto function $\pi_t : \Sigma_t \mapsto \Sigma_{t-1}$ defined for all symbols in

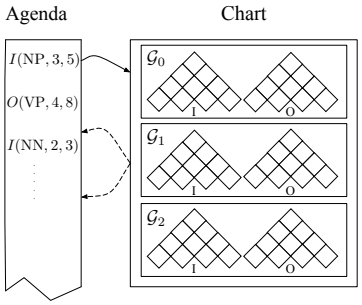


Figure 3: Operation of hierarchical A* parsing. An edge comes off the agenda and is added to the chart (solid line). From this edge, multiple new edges can be constructed and added to the agenda (dashed lines). The chart is composed of two subcharts for each grammar in the hierarchy: an inside chart (I) and an outside chart (O).

$\mathcal{G}_t$; hereafter, we will use A'_t to represent $\pi_t(\mathrm{A}_t)$. A projection is a *relaxation* if, for every rule $r = \mathrm{A}_t \rightarrow \mathrm{B}_t\ \mathrm{C}_t$ with weight w_r the projection $r' = \pi_t(r) = \mathrm{A}'_t \rightarrow \mathrm{B}'_t\mathrm{C}'_t$ has weight $w_{r'} \leq w_r$ in $\mathcal{G}_{t-1}$. Given a target grammar $\mathcal{G}_m$ and a projection function π_m, it is easy to construct a relaxed projection $\mathcal{G}_{m-1}$ by minimizing over rules collapsed by π_m:

$$w_{r'} = \min_{r \in \mathcal{G}_m : \pi_m(r) = r'} w_r$$

Given a series of projection functions $\pi_1 \dots \pi_m$, we can construct relaxed projections by projecting $\mathcal{G}_m$ to $\mathcal{G}_{m-1}$, then $\mathcal{G}_{m-1}$ to $\mathcal{G}_{m-2}$ and so on. Note that by construction, parses in a relaxed projection give lower bounds on parses in the target grammar (Klein and Manning, 2003b).

HA* differs from standard A* in two ways. First, it tracks not only standard inside edges $e = I(\mathrm{X}, i, j)$ which represent derivations of $\mathrm{X} \rightarrow \mathrm{T}^i \dots \mathrm{T}^j$, but also *outside edges* $o = O(\mathrm{X}, i, j)$ which represent derivations of $\mathrm{R} \rightarrow \mathrm{T}^0 \dots \mathrm{T}^{i-1}\ \mathrm{X}\ \mathrm{T}^{j+1} \dots \mathrm{T}^n$. For example, where $I(\mathrm{VP}, 0, 3)$ denotes trees rooted at VP covering the span $[0, 3]$, $O(\mathrm{VP}, 0, 3)$ denotes the derivation of the "rest" of the structure to the root. Where inside edges e have scores β_e which represent (approximations of) their Viterbi inside scores, outside edges o have scores α_o which are (approximations of) their Viterbi outside scores. When we need to denote the inside version of an outside edge, or the reverse, we write $o = \bar{e}$, etc.

Name	Rule				Priority
IN-BASE	$O(\mathrm{T}_t^i, i, i+1) : \alpha_T$	$\Rightarrow$	$I(\mathrm{T}_t^i, i, i+1)$	$: 0$	α_T
IN	$r : w_r \;\; O(\mathrm{A}'_t, i, j) : \alpha_{A'} \;\; I(\mathrm{B}_t, i, k) : \beta_B \;\; I(\mathrm{C}_t, k, j) : \beta_C$	$\Rightarrow$	$I(\mathrm{A}_t, i, j)$	$: \beta_A = \beta_B + \beta_C + w_r$	$\beta_A + \alpha_{A'}$
OUT-BASE	$I(\mathrm{R}_t, 0, n) : \beta_R$	$\Rightarrow$	$O(\mathrm{R}_t, 0, n)$	$: 0$	β_R
OUT-L	$r : w_r \;\; O(\mathrm{A}_t, i, j) : \alpha_A \;\; I(\mathrm{B}_t, i, k) : \beta_B \;\; I(\mathrm{C}_t, k, j) : \beta_C$	$\Rightarrow$	$O(\mathrm{B}_t, i, k)$	$: \alpha_B = \alpha_A + \beta_C + w_r$	$\beta_B + \alpha_B$
OUT-R	$r : w_r \;\; O(\mathrm{A}_t, i, j) : \alpha_A \;\; I(\mathrm{B}_t, i, k) : \beta_B \;\; I(\mathrm{C}_t, k, j) : \beta_C$	$\Rightarrow$	$O(\mathrm{C}_t, k, j)$	$: \alpha_C = \alpha_A + \beta_B + w_r$	$\beta_C + \alpha_C$

Table 2: Deduction rules for HA*. The rule r is in all cases $\mathrm{A}_t \rightarrow \mathrm{B}_t \; \mathrm{C}_t$.

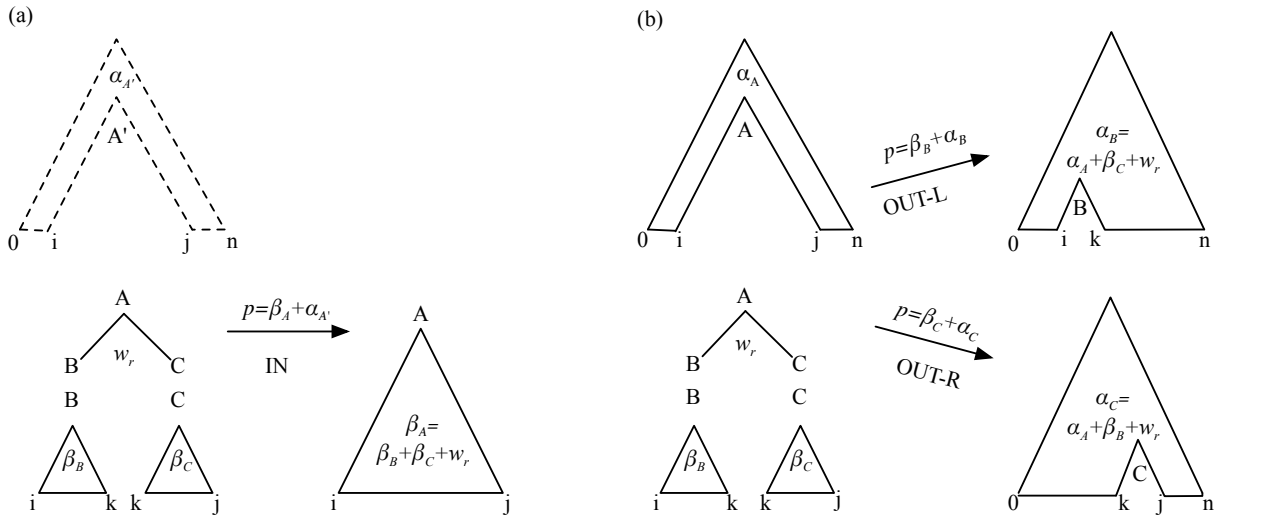


Figure 2: Non-base case deduction rules for HA* depicted graphically. (a) shows the rule used to build inside edges and (b) shows the rules to build outside edges. Inside edges are depicted as complete triangles, while outside edges are depicted as chevrons. An edge from a previous level in the hierarchy is denoted with dashed lines.

The second difference is that HA* tracks items from all levels of the hierarchy on a single, shared agenda, so that all items compete (see Figure 3). While there is only one agenda, it is useful to imagine several charts, one for each type of edge and each grammar level. In particular, outside edges from one level of the hierarchy are the source of completion costs (heuristics) for inside edges at the next level.

The deduction rules for HA* are given in Table 2 and represented graphically in Figure 2. The IN rule (a) is the familiar deduction rule from standard A*: we can combine two adjacent inside edges using a binary rule to form a new inside edge. The new twist is that because heuristics (scores of outside edges from the previous level) are also computed on the fly, they may not be ready yet. Therefore, we cannot carry out this deduction until the required outside edge is present in the previous level's chart. That is, fine inside deductions wait for the relevant coarse outside edges to be popped. While coarse outside edges contribute to *priorities* of refined inside scores (as heuristic values), they do not actually affect the inside scores of edges (again just like basic A*).

In standard A*, we begin with all terminal edges on the agenda. However, in HA*, we cannot enqueue refined terminal edges until their outside scores are ready. The IN-BASE rule specifies the base case for a grammar $\mathcal{G}_t$: we cannot begin until the outside score for the terminal symbol T is ready in the coarser grammar $\mathcal{G}_{t-1}$. The initial queue contains only the most abstract level's terminals, $I(\mathrm{T}_0^i, i, i+1)$. The entire search terminates when the inside edge $I(\mathrm{R}_m, 0, n)$, represting root derivations in the target grammar, is dequeued.

The deductions which assemble outside edges are less familiar from the standard A* algorithm. These deductions take *larger* outside edges and produce *smaller* sub-edges by linking up with inside edges, as shown in Figure 2(b). The OUT-BASE rule states that an outside pass for $\mathcal{G}_t$ can be started if the inside score of the root symbol for that level R_t has been computed. The OUT-L and OUT-R rules are

the deduction rules for building outside edges. OUTL states that, given an outside edge over the span $[i, j]$ and some inside edge over $[i, k]$, we may construct an outside edge over $[k, j]$. For outside edges, the score reflects an estimate of the Viterbi outside score.

As in standard A*, inside edges are placed on the agenda with a priority equal to their path cost (inside score) and some estimate of their completion cost (outside score), now taken from the previous projection rather than a black box. Specifically, the priority function takes the form $p(e) = \beta_e + \alpha_{\bar{e}'}$, where $\bar{e}'$ is the outside version of e one level previous in the hierarchy.

Outside edges also have priorities which combine path costs with a completion estimate, except that the roles of inside and outside scores are reversed: the path cost for an outside edge o is its (outside) score α_o, while the completion cost is some estimate of the inside score, which is the weight β_e of o's complementary edge $e = \bar{o}$. Therefore, $p(o) = \alpha_o + \beta_{\bar{o}}$.

Note that inside edges combine their inside score estimates with outside scores from a *previous level* (a lower bound), while outside edges combine their outside score estimates with inside scores from *the same level*, which are already available. Felzenszwalb and McAllester (2007) show that these choices of priorities have the same guarantee as standard A*: whenever an inside or outside edge comes off the queue, its path cost is optimal.

2.3 Agenda-driven Coarse-to-Fine Parsing

We can always replace the HA* priority function with an alternate priority function of our choosing. In doing so, we may lose the optimality guarantees of HA*, but we may also be able to achieve significant increases in performance. We do exactly this in order to put CTF pruning in an agenda-based framework. An agenda-based implementation allows us to put CTF on a level playing field with HA*, highlighting the effectiveness of the various parsing strategies and normalizing their implementations.

First, we define coarse-to-fine pruning. In standard CTF, we exhaustively parse in each projection level, but skip edges whose projections in the previous level had sufficiently low scores. In particular, an edge e in the grammar $\mathcal{G}_t$ will be skipped entirely if its projection e' in $\mathcal{G}_{t-1}$ had a low *max marginal*: $\alpha_{\bar{e}'} + \beta_{e'}$, that is, the score of the best tree containing e' was low compared to the score best overall root derivation $\beta_{R'}$. Formally, we prune all e where $\alpha_{\bar{e}'} + \beta_{e'} > \beta_{R'} + \tau$ for some threshold τ.

The priority function we use to implement CTF in our agenda-based framework is:

$$p(e) = \beta_e$$

$$p(o) = \begin{cases} \infty & \alpha_o + \beta_{\bar{o}} > \beta_{R_t} + \tau_t \\ \alpha_o + \beta_{\bar{o}} & \text{otherwise} \end{cases}$$

Here, $\tau_t \geq 0$ is a user-defined threshold for level t and β_{R_t} is the inside score of the root for grammar $\mathcal{G}_t$. These priorities lead to uniform-cost exploration for inside edges and completely suppress outside edges which would have been pruned in standard CTF. Note that, by the construction of the IN rule, pruning an outside edge also prunes all inside edges in the next level that depend on it; we therefore prune slightly earlier than in standard CTF. In any case, this priority function maintains the *set* of states explored in CKY-based CTF, but does not necessarily explore those states in the same order.

3 Experiments

3.1 Evaluation

Our focus is parsing speed. Thus, we would ideally evaluate our algorithms in terms of CPU time. However, this measure is problematic: CPU time is influenced by a variety of factors, including the architecture of the hardware, low-level implementation details, and other running processes, all of which are hard to normalize.

It is common to evaluate best-first parsers in terms of edges *popped* off the agenda. This measure is used by Charniak *et al.* (1998) and Klein and Manning (2003b). However, when edges from grammars of varying size are processed on the same agenda, the number of successor edges per edge popped changes depending on what grammar the edge was constructed from. In particular, edges in more refined grammars are more expensive than edges in coarser grammars. Thus, our basic unit of measurement will be edges *pushed* onto the agenda. We found in our experiments that this was well correlated with CPU time.

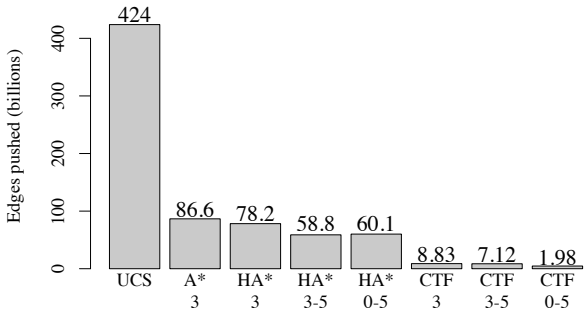


Figure 4: Efficiency of several hierarchical parsing algorithms, across the test set. UCS and all A* variants are optimal and thus make no search errors. The CTF variants all make search errors on about 2% of sentences.

3.2 State-Split Grammars

We first experimented with the grammars described in Petrov *et al.* (2006). Starting with an X-Bar grammar, they iteratively refine each symbol in the grammar by adding latent substates via a split-merge procedure. This training procedure creates a natural hierarchy of grammars, and is thus ideal for our purposes. We used the Berkeley Parser[2] to train such grammars on sections 2-21 of the Penn Treebank (Marcus *et al.*, 1993). We ran 6 split-merge cycles, producing a total of 7 grammars. These grammars range in size from 98 symbols and 8773 rules in the unsplit X-Bar grammar to 1139 symbols and 973696 rules in the 6-split grammar. We then parsed all sentences of length ≤ 30 of section 23 of the Treebank with these grammars. Our "target grammar" was in all cases the largest (most split) grammar. Our parsing objective was to find the Viterbi derivation (i.e. fully refined structure) in this grammar. Note that this differs from the objective used by Petrov and Klein (2007), who use a variational approximation to the most probable parse.

3.2.1 A* versus HA*

We first compare HA* with standard A*. In A* as presented by Klein and Manning (2003b), an auxiliary grammar can be used, but we are restricted to only one and we must compute inside and outside estimates for that grammar exhaustively. For our single auxiliary grammar, we chose the 3-split grammar; we found that this grammar provided the best overall speed.

For HA*, we can include as many or as few auxiliary grammars from the hierarchy as desired. Ideally, we would find that each auxiliary grammar increases performance. To check this, we performed experiments with all 6 auxiliary grammars (0-5 split); the largest 3 grammars (3-5 split); and only the 3-split grammar.

Figure 4 shows the results of these experiments. As a baseline, we also compare with uniform cost search (UCS) (A* with $h = 0$). A* provides a speed-up of about a factor of 5 over this UCS baseline. Interestingly, HA* using only the 3-split grammar is faster than A* by about 10% despite using the same grammars. This is because, unlike A*, HA* need not exhaustively parse the 3-split grammar before beginning to search in the target grammar.

When we add the 4- and 5-split grammars to HA*, it increases performance by another 25%. However, we can also see an important failure case of HA*: using all 6 auxiliary grammars actually *decreases* performance compared to using only 3-5. This is because HA* requires that auxiliary grammars are all relaxed projections of the target grammar. Since the weights of the rules in the smaller grammars are the minimum of a large set of rules in the target grammar, these grammars have costs that are so cheap that all edges in those grammars will be processed long before much progress is made in the refined, more expensive levels. The time spent parsing in the smaller grammars is thus entirely wasted. This is in sharp contrast to hierarchical CTF (see below) where adding levels is always beneficial.

To quantify the effect of optimistically cheap costs in the coarsest projections, we can look at the degree to which the outside costs in auxiliary grammars underestimate the true outside cost in the target grammar (the "slack"). In Figure 5, we plot the average slack as a function of outside context size (number of unincorporated words) for each of the auxiliary grammars. The slack for large outside contexts gets very large for the smaller, coarser grammars. In Figure 6, we plot the number of edges pushed when bounding with each auxiliary grammar individually, against the average slack in that grammar. This plot shows that greater slack leads to more work, reflecting the theoretical property of A* that the work done can be exponential in the slack of the heuristic.

3.2.2 HA* versus CTF

In this section, we compare HA* to CTF, again using the grammars of Petrov *et al.* (2006). It is

[2] http://berkeleyparser.googlecode.com

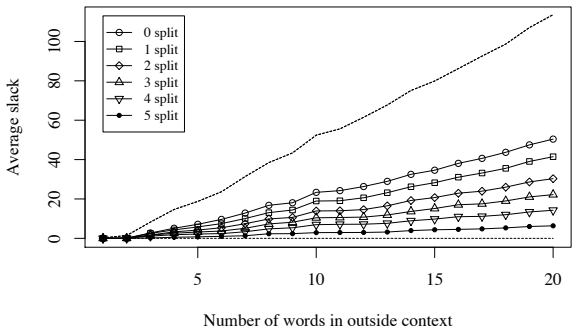


Figure 5: Average slack (difference between estimated outside cost and true outside cost) at each level of abstraction as a function of the size of the outside context. The average is over edges in the Viterbi tree. The lower and upper dashed lines represent the slack of the exact and uniformly zero heuristics.

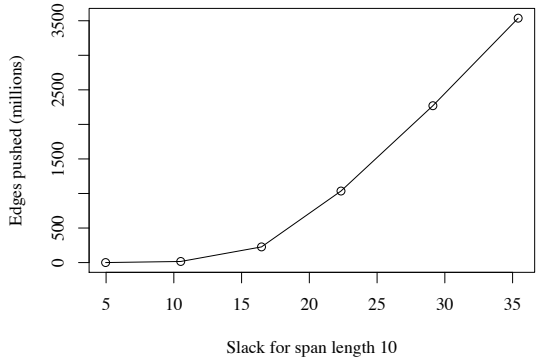


Figure 6: Edges pushed as a function of the average slack for spans of length 10 when parsing with each auxiliary grammar individually.

important to note, however, that we do not use the same grammars when parsing with these two algorithms. While we use the same *projections* to coarsen the target grammar, the scores in the CTF case need not be lower bounds. Instead, we follow Petrov and Klein (2007) in taking coarse grammar weights which make the induced distribution over trees as close as possible to the target in KL-divergence. These grammars represent not a minimum projection, but more of an average.[3]

The performance of CTF as compared to HA* is shown in Figure 4. CTF represents a significant speed up over HA*. The key advantage of CTF, as shown here, is that, where the work saved by using coarser projections falls off for HA*, the work saved with CTF increases with the addition of highly coarse grammars. Adding the 0- through 2-split grammars to CTF was responsible for a factor of 8 speed-up with no additional search errors.

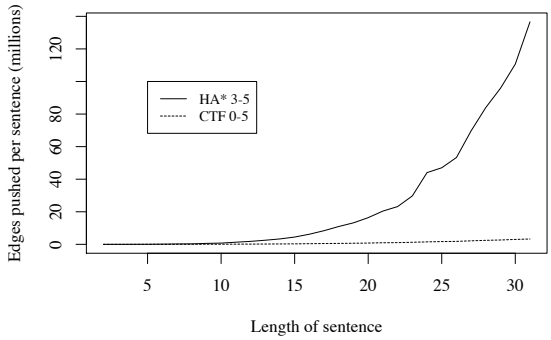


Figure 7: Edges pushed as function of sentence length for HA* 3-5 and CTF 0-5.

Another important property of CTF is that it scales far better with sentence length than does HA*. Figure 7 shows a plot of edges pushed against sentence length. This is not surprising in light of the increase in slack that comes with parsing longer sentences. The more words in an outside context, the more slack there will generally be in the outside estimate, which triggers the time explosion.

Since we prune based on thresholds τ_t in CTF, we can explore the relationship between the number of search errors made and the speed of the parser. While it is possible to tune thresholds for each grammar individually, we use a single threshold for simplicity. In Figure 8, we plot the performance of CTF using all 6 auxiliary grammars for various values of τ. For a moderate number of search errors ($< 5\%$), CTF parses more than 10 times faster than HA* and nearly 100 times faster than UCS. However, below a certain tolerance for search errors ($< 1\%$) on these grammars, HA* is the faster option.[4]

3.3 Lexicalized parsing experiments

We also experimented with the lexicalized parsing model described in Klein and Manning (2003a). This lexicalized parsing model is constructed as the product of a dependency model and the unlexical-

[3] We tried using these average projections as heuristics in HA*, but doing so violates consistency, causes many search errors, and does not substantially speed up the search.

[4] In Petrov and Klein (2007), fewer search errors are reported; this difference is because their search objective is more closely aligned to the CTF pruning criterion.

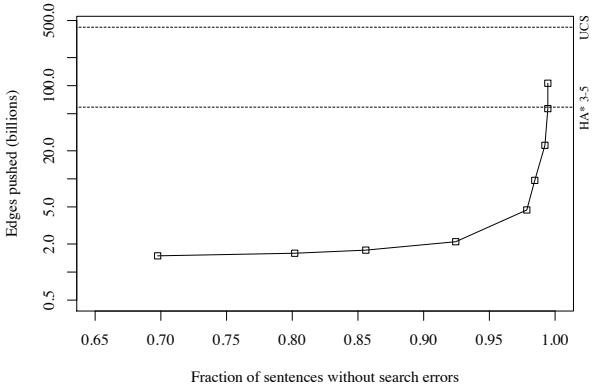


Figure 8: Performance of CTF as a function of search errors for state split grammars. The dashed lines represent the time taken by UCS and HA* which make no search errors. As search accuracy increases, the time taken by CTF increases until it eventually becomes slower than HA*. The y-axis is a log scale.

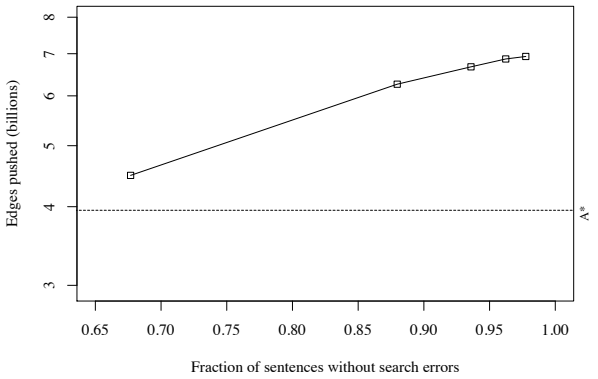


Figure 9: Performance of CTF for lexicalized parsing as a function of search errors. The dashed line represents the time taken by A*, which makes no search errors. The y-axis is a log scale.

ized PCFG model in Klein and Manning (2003c). We constructed these grammars using the Stanford Parser.[5] The PCFG has 19054 symbols 36078 rules. The combined (sentence-specific) grammar has n times as many symbols and $2n^2$ times as many rules, where n is the length of an input sentence. This model was trained on sections 2-20 of the Penn Treebank and tested on section 21.

For these lexicalized grammars, we did not perform experiments with UCS or more than one level of HA*. We used only the single PCFG projection used in Klein and Manning (2003a). This grammar differs from the state split grammars in that it factors into two separate projections, a dependency projection and a PCFG. Klein and Manning (2003a) show that one can use the sum of outside scores computed in these two projections as a heuristic in the combined lexicalized grammar. The generalization of HA* to the factored case is straightforward but not effective. We therefore treated the dependency projection as a black box and used only the PCFG projection inside the HA* framework. When computing A* outside estimates in the combined space, we use the sum of the two projections' outside scores as our completion costs. This is the same procedure as Klein and Manning (2003a). For CTF, we carry out a uniform cost search in the combined space where we have pruned items based on their max-marginals in the PCFG model only.

In Figure 9, we examine the speed/accuracy trade off for the lexicalized grammar. The trend here is the reverse of the result for the state split grammars: HA* is always faster than posterior pruning, even for thresholds which produce many search errors. This is because the heuristic used in this model is actually an extraordinarily tight bound – on average, the slack even for spans of length 1 was less than 1% of the overall model cost.

4 Conclusions

We have a presented an empirical comparison of hierarchical A* search and coarse-to-fine pruning. While HA* does provide benefits over flat A* search, the extra levels of the hierarchy are dramatically more beneficial for CTF. This is because, in CTF, pruning choices cascade and even very coarse projections can prune many highly unlikely edges. However, in HA*, overly coarse projections become so loose as to not rule out anything of substance. In addition, we experimentally characterized the failure cases of A* and CTF in a way which matches the formal results on A*: A* does vastly more work as heuristics loosen and only outperforms CTF when either near-optimality is desired or heuristics are extremely tight.

Acknowledgements

This work was partially supported by an NSERC Post-Graduate Scholarship awarded to the first author.

[5] http://nlp.stanford.edu/software/

References

Sharon Caraballo and Eugene Charniak. 1996. Figures of Merit for Best-First Probabalistic Parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Eugene Charniak. 1997 Statistical Parsing with a Context-Free Grammar and Word Statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*.

Eugene Charniak, Sharon Goldwater and Mark Johnson. 1998. Edge-based Best First Parsing. In *Proceedings of the Sixth Workshop on Very Large Corpora*.

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. 2006. Multilevel Coarse-to-fine PCFG Parsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Michael Collins. 1997. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

P. Felzenszwalb and D. McAllester. 2007. The Generalized A* Architecture. In *Journal of Artificial Intelligence Research*.

Aria Haghighi, John DeNero, and Dan Klein. 2007. Approximate Factoring for A* Search. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Dan Klein and Chris Manning. 2002. Fast Exact Inference with a Factored Model for Natural Language Processing. In *Advances in Neural Information Processing Systems*.

Dan Klein and Chris Manning. 2003. Factored A* Search for Models over Sequences and Trees. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Dan Klein and Chris Manning. 2003. A* Parsing: Fast Exact Viterbi Parse Selection. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*

Dan Klein and Chris Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

M. Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics.*

Mark-Jan Nederhof. 2003. Weighted deductive parsing and Knuth's algorithm. In *Computational Linguistics*, 29(1):135–143.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2003. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. In *Journal of Logic Programming*, 24:3–36.

An effective Discourse Parser that uses Rich Linguistic Information

Rajen Subba *
Display Advertising Sciences
Yahoo! Labs
Sunnyvale, CA, USA
rajen@yahoo-inc.com

Barbara Di Eugenio
Department of Computer Science
University of Illinois
Chicago, IL, USA
bdieugen@cs.uic.edu

Abstract

This paper presents a first-order logic learning approach to determine rhetorical relations between discourse segments. Beyond linguistic cues and lexical information, our approach exploits compositional semantics and segment discourse structure data. We report a statistically significant improvement in classifying relations over attribute-value learning paradigms such as Decision Trees, RIPPER and Naive Bayes. For discourse parsing, our modified shift-reduce parsing model that uses our relation classifier significantly outperforms a right-branching majority-class baseline.

1 Introduction

Many theories postulate a hierarchical structure for discourse (Mann and Thompson, 1988; Moser et. al., 1996; Polanyi et. al., 2004). Discourse structure is most often based on semantic / pragmatic relationships between spans of text and results in a tree structure, as that shown in Figure 1. Discourse parsing, namely, deriving such tree structures and the *rhetorical relations* labeling their inner nodes is still a challenging and mostly unsolved problem in NLP. It is linguistically plausible that such structures are determined at least in part on the basis of the meaning of the related chunks of texts, and of the rhetorical intentions of their authors. However, such knowledge is extremely difficult to capture. Hence, previous work on discourse parsing (Wellner et. al., 2006; Sporleder and Lascarides, 2005; Marcu, 2000; Polanyi et. al., 2004; Soricut and Marcu, 2003; Baldridge and Lascarides, 2005) has relied only on syntactic and lexical information, lexical chains and shallow semantics.

We present an innovative discourse parser that uses compositional semantics (when available) and information on the structure of the segment being built itself. Our discourse parser, based on a modified shift-reduce algorithm, crucially uses a rhetorical relation classifier to determine the site of attachment of a new incoming chunk together with the appropriate relation label. Another novel aspect of our work is the usage of Inductive Logic Programming (ILP): ILP learns from first-order logic representations (FOL). The ILP-based relation classifier is significantly more accurate than relation classifiers that use competitive propositional ML algorithms such as decision trees and Naive Bayes. In addition, it results in FOL rules that are linguistically perspicuous. Our domain is that of instructional how-to-do manuals, and we describe our corpus in Section 2. In Section 3, we discuss the modified shift-reduce parser we developed. The bulk of the paper is devoted to the rhetorical relation classifier in Section 4. Experimental results of both the relation classifier and the discourse parser in its entirety are discussed in Section 5. Further details can be found in (Subba, 2008).

2 Discourse Annotated Instructional Corpus

Existing corpora annotated with rhetorical relations (Carlson et. al., 2003; Wolf and Gibson, 2005; Prasad et. al., 2008) focus primarily on news articles. However, for us the development of the discourse parser is parasitic on our ultimate goal: developing resources and algorithms for language in-

*This work was done while the author was a student at the University of Illinois at Chicago.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 566–574,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

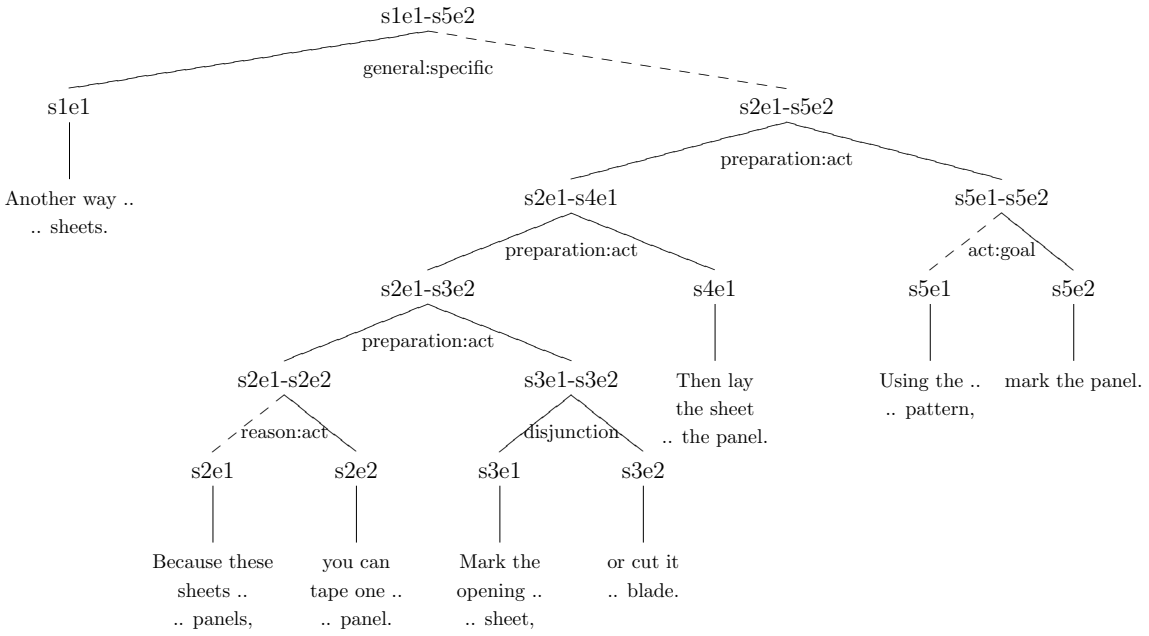


Figure 1: Discourse Parse Tree of the Text in Example (1)

terfaces to instructional applications. Hence, we are interested in working with instructional texts. We worked with a corpus on home-repair that is about 5MB in size and is made up entirely of written English instructions,[1] such as that shown in Example (1). The text has been manually segmented into Elementary Discourse Units (EDUs), the smallest units of discourse. In total, our corpus contains 176 documents with an average of 32.6 EDUs for a total of 5744 EDUs and 53,250 words. The structure for Example (1) is shown in Figure 1.

(1) [Another way to measure and mark panels for cutting is to make a template from the protective sheets.$_{(s1e1)}$] [Because these sheets are the same size as the panels,$_{(s2e1)}$] [you can tape one to the wall as though it were a panel.$_{(s2e2)}$] [Mark the opening on the sheet$_{(s3e1)}$] [or cut it out with a razor blade.$_{(s3e2)}$] [Then lay the sheet on the panel.$_{(s4e1)}$] [Using the template as a pattern,$_{(s5e1)}$] [mark the panel.$_{(s5e2)}$]

To explore our hypothesis, that rich linguistic information helps discourse parsing, and that the state of the art in machine learning supports such an approach, we needed training data annotated with both compositional semantics and rhetorical relations. We performed the first type of annotation almost completely automatically, and the second manually, as we turn now to describing.

2.1 Compositional Semantics Derivation

The type of compositional semantics we are interested in is heavily rooted in verb semantics, which is particularly appropriate for the instructional text we are working with. Therefore, we used VerbNet (Kipper et. al., 2000) as our verb lexicon. VerbNet groups together verbs that undergo the same syntactic alternations and share similar semantics. It accounts for 4962 distinct verbs classified into 237 main classes. Each verb class is described by thematic roles, selectional restrictions on the arguments and frames consisting of a syntactic description and semantic predicates. Such semantic classification of verbs can be helpful in making generalizations, especially when data is not abundant. Generalization can also be achieved by means of the semantic predicates. Although the verb classes of two verb instances may differ, semantic predicates are shared across verbs. To compositionally build verb based

[1] The raw corpus was originally assembled at the Information Technology Research Institute, University of Brighton.

semantic representations of our EDUs, we (Subba et al., 2006) integrated a robust parser, LCFLEX (Rosé, 2000), with a lexicon and ontology based both on VerbNet and, for nouns, on CoreLex (Buitelaar, 1998). The augmented parser was able to derive complete semantic representations for 3257 of the 5744 EDUs (56.7%). The only manual step was to pick the correct parse from a forest of parse trees, since the output of the parser can be ambiguous.

2.2 Rhetorical relation annotation

The discourse processing community has not yet reached agreement on an inventory of rhetorical relations. Among the many choices, our coding scheme is a hybrid of (Moser et. al., 1996) and (Marcu, 1999). We focused on what we call *informational relations*, namely, relations in the domain. We used 26 relations, divided into 5 broad classes: 12 **causal** relations (e.g., *preparation:act, goal:act, cause:effect, step1:step2*); 6 **elaboration** relations (e.g., *general:specific, set:member, object:attribute*; 3 **similarity** relations (*contrast1:contrast2, comparison, restatement*); 2 **temporal** relations (*co-temp1:co-temp2, before:after*); and 4 **other** relations, including *joint* and *disjunction*.

The annotation yielded 5172 relations, with reasonable intercoder agreement. On 26% of the data, we obtained $\kappa = 0.66$; κ rises to 0.78 when the two most commonly confused relations, *preparation:act* and *step1:step2*, are consolidated. We also annotated the relata as *nucleus* (more important member) and *satellite* (contributing member(s)) (Mann and Thompson, 1988), with $\kappa = 0.67$.[2] The most frequent relation is *preparation:act* (24.46%), and in general, causal relations are more frequently used in our instructional corpus than in news corpora (Carlson et. al., 2003; Wolf and Gibson, 2005).

3 Shift-Reduce Discourse Parsing

Our discourse parser is a modified version of a shift-reduce parser. The shift operation places the next segment on top of the stack, TOP. The reduce operation will attach the text segment at TOP to the text segment at TOP-1. (Marcu, 2000) also uses a shift-reduce parser, though our parsing algorithm differs in two respects: 1) we do not learn shift operations and 2) in contrast to (Marcu, 2000), the attachment of an incoming text segment to the emerging tree may occur at any node on the right frontier. This allows for the more sophisticated type of adjunction operations required for discourse parsing as modeled in D-LTAG (Webber, 2004). A reduce operation is determined by the relation identification component. We check if a relation exists between the incoming text segment and the attachment points on the right frontier. If more than one attachment site exists, then the attachment site for which the rule with the highest score fired (see below) is chosen for the *reduce* operation. A reduce operation can further trigger additional reduce operations if there is more than one tree left in the stack after the first reduce operation. When no rules fire, a *shift* occurs. In the event that all the segments in the input list have been processed and a full DPT has not been obtained, then we reduce TOP and TOP-1 using the *joint* relation until a single DPT is built.

4 Classifying Rhetorical Relations

Identifying the informational relations between text segments is central to our approach for building the informational tree structure of text. We believe that the use of a limited knowledge representation formalism, essentially propositional logic, is not adequate and that a relational model that can handle compositional semantics is necessary. We cast the problem of determining informational relations as a classification task. We used the ILP system Aleph that is based on (Muggleton, 1995). Formulation of any problem within the ILP framework consists of background knowledge **B** and the set of examples **E** ($E^+ \cup E^-$). In our ILP framework, positive examples are ground clauses describing a relation and its relata, e.g. *relation(s5e1,s5e2,act:goal)*, or *relation(s2e1-s3e2,s4e1,preparation:act)* from Figure 1. If e is a positive example of a relation r, then it is also a negative example for all the other relations.

Background Knowledge (**B**) can be thought of as features used by ILP to learn rules, as in traditional attribute-value learning algorithms. We use the following information to learn rules for classifying relations. Figure 2 shows a sample of the background

[2] We don't have space to explain why we annotate for nucleus and satellite, even if (Moser et. al., 1996) argue that this sort of distinction does not apply to informational relations.

Verbs + Nouns:	verb('$s5e2$',mark). noun('$s5e2$',panel).
Linguistic Cues:	firstWordPOS('$s5e2$','VB'). lastWordPOS('$s5e2$','.').
Similarity:	segment_ sim_ score('$s5e1$','$s5e2$',0.0).
Compositional Semantics:	verbclass('$s5e2$',mark,'image_impression-25.1'). agent('$s5e2$',frame(mark),you). destination('$s5e2$',frame(mark),panel). cause('$s5e2$',frame(mark),you,'$s5e2$-mark-e'). prep('$s5e2$',frame(mark),end('$s5e2$-mark-e'),mark,panel). created_image('$s5e2$',frame(mark),result('$s5e2$-mark-e'),mark).
Structural Information:	same_sentence('$s5e1$','$s5e2$').

Figure 2: Example Background Knowledge

knowledge provided for EDU $s5e2$.

Verbs + Nouns: These features were derived by tagging all the sentences in the corpus with a POS tagger (Brill, 1995).

WordNet: For each noun in our data, we also use information on hypernymy and meronymy relations using WordNet. In a sense, this captures the domain relations between objects in our data.

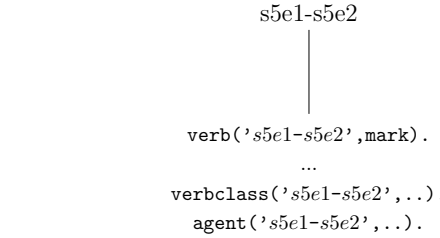
Linguistic Cues: Various cues can facilitate the inference of informational relations, even if it is well known that they are based solely on the content of the text segments, various cues can facilitate the inference of such relations. At the same time, it is well known that relations are often non signalled: in our corpus, only 43% of relations are signalled, consistently with figures from the literature (44% in (Williams and Reiter, 2003) and 45% in (Prasad et. al., 2008)). Besides lexical cues such as *but*, *and* and *if*, we also include modals, tense, comparatives and superlatives, and negation. E.g., *wrong-act* in relations like *prescribe-act:wrong-act* is often expressed using a negation.

Similarity: For the two segments in question, we compute the cosine similarity of the segments using only nouns and verbs.

Compositional semantics: the semantic information derived by our parser, as described in Section 2.1. The semantic representation of segment $s5e2$ from Example (1) is shown in Figure 2. Each semantic predicate is a feature for the classifier.

Structural Information: For relations between two EDUs, we use knowledge of whether the two EDUs are intra-sentential or inter-sentential, since some relations, e.g. *criterion:act*, are more likely to be realized intra-sententially than inter-sententially. For larger segments, we also encode the hierarchical representation of text segments that contain more than one nucleus, the distance between the nuclei of the two segments and any relations that exist between the smaller inner segments.

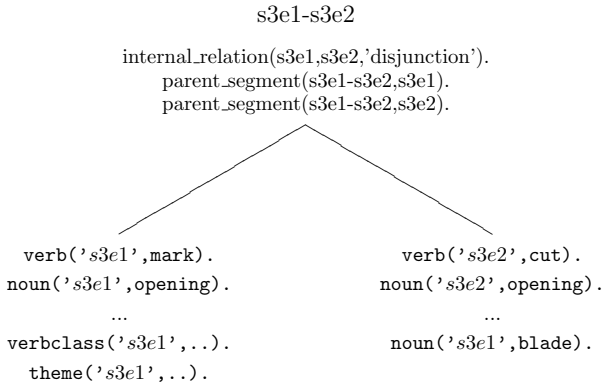
At this point, the attentive reader will be wondering how we encode compositional semantics for relations relating text segments larger than one EDU. Clearly we cannot just list the semantics of each EDU that is dominated by the larger segment. We follow the intuition that nuclei represent the most important portions of segments (Mann and Thompson, 1988). For segments such as $s5e1$-$s5e2$ that contains a single nucleus, we simply reduce the semantic content of the larger segment to that of its nucleus:



In this case, the semantics of the complex text segment is represented by the compositional semantics of the single most important EDU.

For segments that contain more than one nucleus, such as $s3e1$-$s3e2$, the discourse structure information of the segment is represented with the additional predicates *internal_relation* and *parent_segment*. These predicates can be used recursively at every level of the tree to specify the relation between the most important segments. In addition, they also provide a means to represent the compositional semantics of the most important EDUs and

make them available to the relational learning algorithm.



4.1 Learning FOL Rules for Discourse Parsing

In Aleph, the hypothesis space is restricted to a set of rules that conform to a predefined language *L*. This is done with the use of *mode declarations* which, in other words, introduces a language bias in the learning process. *modeh* declarations inform the learning algorithm about what predicates to use as the head of the rule and *modeb* specifies what predicates to use in the body of the rule. Not all the information in **B** needs to be included in the body of the rule. This makes sense since we often learn definitions of concepts based on more abstract higher level information that is inferred from some other information that is not part of our final definition. Mode declarations are used by Aleph to build the most specific clause ($\perp$) that can be learned for each example. $\perp$ constrains the search for suitable hypotheses. $\perp_i$ is built by taking an example $e_i \in \boldsymbol{E}^+$ and adding literals that are entailed by **B** and e_i. We then have the following property, where H_i is the hypothesis (rule) we are trying to learn and $\preceq$ is a generality operator:

$$\square \preceq H_i \preceq \perp_i$$

Finding the most specific clause ($\perp$) provides us with a partially ordered set of clauses from which to choose the best hypothesis based on some quantifiable qualitative criteria. This sub-lattice is bounded by the most general clause ($\square$, the empty clause) from the top and the most specific clause ($\perp$) at the bottom. We use the heuristic search in Aleph that is similar to the A*-like search strategy presented by (Muggleton, 1995) to find the best hypothesis (rule). A noise threshold on the number of negative examples that can be covered by a rule can be set. We learn a model that learns perfect rules first and then one that allows for at most 5 negative examples. A backoff model that first uses the model trained with $noise = 0$ and then $noise = 5$ if no classification has been made is used. We use the evaluation function in Equation 1 to guide our search through the tree of possible hypotheses. This evaluation function is also called the compression function since it prefers simpler explanations to more complex ones (Occam's Razor). f_s is the score for clause c_s that is being evaluated, p_s is the number of positive examples, n_s is the number of negative examples, l_s is the length of the clause (measured by the number of clauses).

$$f_s = p_s - (n_s + (0.1 \times l_s)) \quad (1)$$

Classification in most ILP systems, including Aleph, is restricted to binary classification (positive vs. negative). In many applications with just two classes, this is sufficient. However, we are faced with a multi-classification problem. In order to perform multi-class classification, we use a decision list. First, we build *m* binary classifiers for each relation $r \in R$. Then, we form an ordered list of the rules based on the following criterion:

1. Given two rules r_i and r_j, r_i ,is ranked higher than r_j if $(\mathrm{p}_i - n_i) > (\mathrm{p}_j - n_j)$.
2. if $(\mathrm{p}_i - n_i) = (\mathrm{p}_j - n_j)$, then r_i is ranked higher than r_j if $(\frac{p_i}{p_i+n_i}) > (\frac{p_j}{p_j+n_j})$.
3. if $(\mathrm{p}_i - n_i) = (\mathrm{p}_j - n_j)$ and $(\frac{p_i}{p_i+n_i}) = (\frac{p_j}{p_j+n_j})$ then r_i is ranked higher than r_j if $(\mathrm{l}_i) > (\mathrm{l}_j)$.
4. default: random order

Classifying an unseen example is done by using the first rule in the ordered list that satisfies it.

5 Experiments and Results

We report our results from experiments on both the classification task and the discourse parsing task.

5.1 Relation Classification Results

For the classification task, we conducted experiments using the stratified k-fold (k = 5) cross-validation evaluation technique on our data. Unlike

(Wellner et. al., 2006; Sporleder and Lascarides, 2005), we do not assume that we know the order of the relation in question. Instead we treat reversals of non-commutative relations (e.g. *preparation:act* and *act:goal*) as separate relations as well. We compare our ILP model to RIPPER, Naive Bayes and the Decision Tree algorithm. We should point out that since attribute-value learning models cannot handle first-order logic data, they have been presented with features that lose at least some of this information. While this may then seem to result in an unfair comparison, to the contrary, this is precisely the point: can we do better than very effective attribute-value approaches that however inherently cannot take richer information into account? All the statistical significance tests were performed using the value of F-Score obtained from each of the folds. We report performance on two sets of data since we were not able to obtain compositional semantic data for all the EDUs in our corpus:

- Set A: Examples for which semantic data was available for all the nuclei of the segments (1789 total). This allows us to have a better idea of how much impact semantic data has on the performance, if any.
- Set B: All examples regardless of whether or not semantic data was available for the nuclei of the segments (5475 total).

Model	Semantics	No Semantics
ILP	**62.78**	60.25
Decision Tree	56.29	55.45
RIPPER	58.02	56.96
Naive Bayes	35.83	34.66
Majority Class	31.63	31.63

Table 1: Classification Performance: Set A (F-Score)

Table 1 shows the results on Set A. ILP outperforms all the other models. Via ANOVA, we first conclude that there is a statistically significant difference between the 8 models ($p < 2.2e^{-16}$). To then pinpoint where the difference precisely lies, pairwise comparisons using Student's t-test show that the difference between ILP (using semantics) and all of the other learning models is statistically significant at $p < 0.05$. Additionally, ILP with semantics is significantly better than ILP without it ($p < 0.05$). For Decision Tree, Naive Bayes and RIPPER, the improvement in using semantics is not statistically significant.

Model	Semantics	No Semantics
ILP	**59.43**	59.22
Decision Tree	53.84	53.69
RIPPER	51.1	51.36
Naive Bayes	49.69	51.62
Majority Class	22.01	22.01

Table 2: Classification Performance: Set B (F-Score)

In Table 2, we list the results on Set B. Once again, our ILP model outperforms the other three learning models. Naive Bayes is much more competitive when using all the examples compared to using only examples with semantic data. In the case of the attribute-value machine learning models, the use of semantic data seems to marginally hurt the performance of the classifiers. However, this is in contrast to the relational ILP model which always performs better when using semantics. This result suggests that the use of semantic data with loss of information may not be helpful, and in fact, it may actually hurt performance. Based on ANOVA, the differences in these 8 models is statistically significant with $p < 6.95e^{-12}$. A pairwise t-test between ILP (using semantics) and each of the other attribute-value learning models shows that our results are statistically significant at $p < 0.05$.

In Table 3, we report the performance of the two ILP models on each relation.[3] In general, the models perform better on relations that have the most examples.

The evaluation of work in discourse parsing is hindered by the lack of a standard corpus or task. Hence, our results cannot be directly compared to (Marcu, 2000; Sporleder and Lascarides, 2005; Wellner et. al., 2006), but neither can those works be compared among themselves, because of differences in underlying corpora, the type and number of relations used, and various assumptions. However, we can still draw some general comparisons. Our ILP-based models provide as much or significantly

[3] Due to space limitations, only relations with > 10 examples are shown.

relation	Semantics	No Semantics
preparation:act	74.86	72.05
general:specific	31.74	28.24
joint	55.23	52
act:goal	86.12	83.85
criterion:act	77.37	75.32
goal:act	73.43	68.9
step1:step2	28.75	35.29
co-temp1:co-temp2	48.84	37.84
disjunction	83.33	80.81
act:criterion	54.29	54.79
contrast1:contrast2	22.22	5.0
act:preparation	65.31	70.59
act:reason	0	10.26
cause:effect	19.05	10.53
comparison	22.22	10.53

Table 3: Classification Performance (F-Score) by Relation: ILP on Set A

more improvement over a majority-class baseline when compared to these other works. This is the case even though our work is based on less training data, relatively more relations, relations both between just two EDUs and those involving larger text segments, and we make no assumptions about the order of the relations. Our results are comparable to (Marcu, 2000), which reports an accuracy of about 61% for his classifier. His majority class baseline performs at about 50% accuracy. (Wellner et. al., 2006) reports an accuracy of up to 81%, with a majority class baseline performance of 45.7%. However, our task is more challenging than (Wellner et. al., 2006). They use only 11 relations compared to the 26 we use. They also assume the order of the relation in the examples (i.e. examples for *goal:act* would be treated as examples for *act:goal* by reversing the order of the arguments) whereas we do not make such assumptions. In addition, their training data is almost twice as large as ours, based on relation instances. (Sporleder and Lascarides, 2005) also makes the same assumption on the ordering of the relations as (Wellner et. al., 2006). They report an accuracy of 57.75%. Their work, though, was based on only 5 relations. Importantly, neither (Wellner et. al., 2006; Sporleder and Lascarides, 2005) model examples with complex text segments with more than one EDU.

5.2 How interesting are the rules?

Given that our ILP models learn first-order logic rules, we can make some qualitative analysis of the rules learned, such as those below, learnt by the ILP model that uses semantics:

```
(2a) relation(A,B,'act:goal') :-
     firstWordPOS(A,'VBG'),
     verbclass(A,D,'use-1'),
     firstWordPOS(B,'VB').
     [pos cover = 23 neg cover = 1]

(2b) relation(A,B,'preparation:act') :-
     discourse_cue(B,front,and),
     cause(A,frame(C),D,E),
     theme(B,frame(F),G), theme(A,frame(C),G).
     [pos cover = 12 neg cover = 0]

(2c) relation(A,B,'preparation:act') :-
     discourse_cue(B,front,then),
     parent_segment(A,C), parent_segment(A,D),
     internal_relation(C,D,'preparation:act').
     [pos cover = 17 neg cover = 0]
```

(2a) is learned using examples such as *relation(*$s5e1$,$s5e2$*,'act:goal')* from Example (1). (2b) uses relational semantic information. This rule can be read as follows:

> **IF** segment A contains a cause and a *theme*, the same object that is the *theme* in A is also the *theme* in segment B, and B contains the discourse cue *and* at the front **THEN** the relation between A and B is *preparation:act*.

(2c) is a rule that makes use of the structural information about complex text segments. When using Set A, more than about 60% of the rules induced include at least one semantic predicate in its body. They occur more frequently in rules for relations like *preparation:act* while less in rules for *general:specific* and *act:goal*.

5.3 Discourse Parsing Results

In order to test our discourse parser, we used 151 documents for training and 25 for testing. We evaluated the performance of our parser on both the discourse parse trees it builds at the sentence level and at the document level. The test set contained

		Sentence Level			Document Level		
model	Semantics	span	nuclearity	relation	span	nuclearity	relation
SR-ILP	yes	92.91	71.83	**63.06**	**70.35**	**49.47**	**35.44**
SR-ILP	no	91.98	69.59	58.58	68.95	48.16	33.33
Baseline	-	**93.66**	**74.44**	34.32	70.26	47.98	22.46

Table 4: Parsing Performance (F-Score): (Baseline = right-branching majority)

341 sentences out of which 180 sentences were segmented into more than one EDU. We ran experiments using our two ILP models for the relation identifier, namely ILP with semantics and without semantics. Our ILP based discourse parsing models are named SR-ILP. We compare the performance of our models against a right branching majority class baseline. We used the sign-test to determine statistical significance of the results. Using the automatic evaluation methodology in (Marcu, 2000), precision, recall and F-Score measures are computed for determining the hierarchical spans, nucleus-satellite assignments and rhetorical relations. The performance on labeling relations is the most important measure since the results on nuclearity and hierarchical spans are by-products of the decisions made to attach segments based on relations.

On labeling relations, the parser that uses all the features (including compositional semantics) for determining relations performs the best with an F-Score of 63.06%. The difference of about 4.5% (between ILP with semantics and without semantics) in F-Score is statistically significant at $p = 0.006$. Our best model, SR-ILP (using semantics) beats the baseline by about 28% in F-Score. Since the task at the document level is much more challenging than building the discourse structure at the sentence level, we were not surprised to see a considerable drop in performance. For our best model, the performance on labeling relations drops to 35.44%. Clearly, the mistakes made when attaching segments at lower levels have quite an adverse effect on the overall performance. A less greedy approach to parsing discourse structure is warranted.

While we would have hoped for a better performance than 35.44%, to start with, (Forbes et. al., 2001), (Polanyi et. al., 2004), and (Cristea, 2000) do not report the performance of their discourse parsers at all. (Marcu, 2000) reports precision and recall of up to 63.2% and 59.8% on labeling relations using manually segmented EDUs on three WSJ articles. (Baldridge and Lascarides, 2005) reports 43.2% F-Score on parsing 10 dialogues using a probabilistic head-driven parsing model.

6 Conclusions

In conclusion, we have presented a relational approach for classifying informational relations and a modified shift-reduce parsing algorithm for building discourse parse trees based on informational relations. To our knowledge, this is the first attempt at using a relational learning model for the task of relation classification, or even discourse parsing in general. Our approach is linguistically motivated. Using ILP, we are able to account for rich compositional semantic data of the EDUs based on VerbNet as well as the structural relational properties of the text segments. This is not possible using attribute-value based models like Decision Trees and RIPPER and definitely not using probabilistic models like Naive Bayes. Our experiments have shown that semantics can be useful in classifying informational relations. For parsing, our modified shift-reduce algorithm using the ILP relation classifier outperforms a right-branching baseline model significantly. Using semantics for parsing also yields a statistically significant improvement. Our approach is also domain independent as the underlying model and data are not domain specific.

Acknowledgments

This work is supported by the National Science Foundation (IIS-0133123 and ALT-0536968) and the Office of Naval Research (N000140010640).

References

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

Baldridge, J. and Lascarides, A.: Probabilistic Head-Driven Parsing for Discourse Structure In Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNNL), Ann Arbor, 2005.

Brill, E.: Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. Computational Linguistics, 21(4):543565, 1995.

Buitelaar, P.: CoreLex: Systematic Polysemy and Underspecification. Ph.D. Thesis, Brandies University, 1998.

Carlson, L. D. M. and Okurowski., M. E.: Building a discourse-tagged corpus in the framework of rhetorical structure theory. In Current Directions in Discourse and Dialogue pages 85–112, 2003.

Cristea, D.: An Incremental Discourse Parser Architecture. In D. Christodoulakis (Ed.) Proceedings of the Second International Conference - Natural Language Processing - Patras, Greece, June 2000.

Forbes, K., Miltsakaki, E., R. P. A. S. A. J. and Webber., B.: D-ltag system - discourse parsing with a lexicalized tree adjoining grammar. Information Stucture, Discourse Structure and Discourse Semantics, ESSLLI 2001.

Grosz, B. J. and Sidner, C. L.: Attention, intention and the structure of discourse. Computational Linguistics 12:175–204, 1988.

Hobbs, J. R.: On the coherence and structure of discourse. In Polyani, Livia editor, The Structure of Discourse, 1985.

Kipper, K., H. T. D. and Palmer., M.: Class-based construction of a verb lexicon. AAAI-2000, Proceedings of the Seventeenth National Conference on Artificial Intelligence, 2000.

Mann, W. and Thompson, S.: Rhetorical structure theory: Toward a functional theory of text organization. Text, 8(3):243–281, 1988.

Marcu, D.: Instructions for Manually Annotating the Discourse Structures of Texts. Technical Report, University of Southern California, 1999.

Marcu, D.: The theory and practice of discourse parsing and summarization. Cambridge, Massachusetts, London, England, MIT Press, 2000.

Moser, M. G., Moore, J. D., and Glendening, E.: Instructions for Coding Explanations: Identifying Segments, Relations and Minimal Units. University of Pittsburgh, Department of Computer Science, 1996.

Muggleton, S. H.: Inverse entailment and progol. In New Generation Computing Journal 13:245–286, 1995.

Polanyi, L., Culy, C., van den Berg, M. H. and Thione, G. L.: A Rule Based Approach to Discourse Parsing. Proceedings of the 5th SIGdial Workshop in Discourse And Dialogue. Cambridge, MA USA pp. 108-117., May 1, 2004.

Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L., Joshi, A., and Webber, B.: The Penn Discourse Treebank 2.0. LREC, 2008.

Rosé, C. P.: A Syntactic Framework for Semantic Interpretation, Proceedings of the ESSLLI Workshop on Linguistic Theory and Grammar Implementation, 2000.

Sporleder, C. and Lascarides., A.: Exploiting linguistic cues to classify rhetorical relations. Recent Advances in Natural Language Processing, 2005.

Soricut, R. and Marcu., D.: Sentence level discourse parsing using syntactic and lexical information. Proceedings of the Human Language Technology and North American Assiciation for Computational Linguistics Conference, 2003.

Subba, R., Di Eugenio, B., E. T.: Building lexical resources for princpar, a large coverage parser that generates principled semantic representations. LREC, 2006.

Subba, R.: Discourse Parsing: A Relational Learning Approach Ph.D. Thesis, University of Illinois Chicago, December 2008.

Webber, B.: DLTAG: Extending Lexicalized TAG to Discourse. Cognitive Science 28:751-779, 2004.

Wellner, B., Pustejovsky, J., C. H. R. S. and Rumshisky., A.: Classification of discourse coherence relations: An exploratory study using multiple knowledge sources. In Proceedings of the 7th SIGDIAL Workshop on Discourse and Dialogue, 2006.

Williams, S. and Reiter, E.: A corpus analysis of discourse relations for natural language generation. Proceedings of Corpus Linguistics, pages 899–908, 2003.

Wolf, F. and Gibson, E.: Representing discourse coherence: A corpus-based analysis. Computational Linguistics 31(2):249–287, 2005.

Graph-Cut-Based Anaphoricity Determination for Coreference Resolution

Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
vince@hlt.utdallas.edu

Abstract

Recent work has shown that explicitly identifying and filtering non-anaphoric mentions prior to coreference resolution can improve the performance of a coreference system. We present a novel approach to this task of anaphoricity determination based on graph cuts, and demonstrate its superiority to competing approaches by comparing their effectiveness in improving a learning-based coreference system on the ACE data sets.

1 Introduction

Coreference resolution is the problem of identifying which noun phrases (NPs, or *mentions*) refer to the same real-world entity in a text or dialogue. According to Webber (1979), coreference resolution can be decomposed into two complementary tasks: "(1) identifying what a text potentially makes available for anaphoric reference and (2) constraining the candidate set of a given anaphoric expression down to one possible choice." The first task is nowadays typically formulated as an *anaphoricity determination* task, which aims to classify whether a given mention is anaphoric or not. Knowledge of anaphoricity could improve the precision of a coreference system, since non-anaphoric mentions do not have an antecedent and therefore do not need to be resolved.

Previous work on anaphoricity determination can be broadly divided into two categories (see Poesio et al. (2004) for an overview). Research in the first category aims to identify specific types of non-anaphoric phrases, with some identifying pleonastic *it* (using heuristics [e.g., Paice and Husk (1987), Lappin and Leass (1994), Kennedy and Boguraev (1996)], supervised approaches [e.g., Evans (2001), Müller (2006), Versley et al. (2008)], and distributional methods [e.g., Bergsma et al. (2008)]), and others identifying non-anaphoric definite descriptions (using rule-based techniques [e.g., Vieira and Poesio (2000)] and unsupervised techniques [e.g., Bean and Riloff (1999)]).

On the other hand, research in the second category focuses on (1) determining the anaphoricity of *all* types of mentions, and (2) using the resulting anaphoricity information to improve coreference resolution. For instance, Ng and Cardie (2002a) train an anaphoricity classifier to determine whether a mention is anaphoric, and let an independently-trained coreference system resolve only those mentions that are classified as anaphoric. Somewhat surprisingly, they report that using anaphoricity information adversely affects the performance of their coreference system, as a result of an overly *conservative* anaphoricity classifier that misclassifies many anaphoric mentions as non-anaphoric. One solution to this problem is to use anaphoricity information as *soft* constraints rather than as hard constraints for coreference resolution. For instance, when searching for the best partition of a set of mentions, Luo (2007) combines the probabilities returned by an anaphoricity model and a coreference model to score a coreference partition, such that a partition is penalized whenever an anaphoric mention is resolved. Another, arguably more popular, solution is to "improve" the output of the anaphoricity classifier by exploiting the dependency between anaphoricity determination and coreference resolu-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 575–583,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

tion. For instance, noting that Ng and Cardie's anaphoricity classifier is too conservative, Ng (2004) first parameterizes their classifier such that its conservativeness can be varied, and then tunes this parameter so that the performance of the coreference system is maximized. As another example, Denis and Baldridge (2007) and Finkel and Manning (2008) perform joint inference for anaphoricity determination and coreference resolution, by using Integer Linear Programming (ILP) to enforce the consistency between the output of the anaphoricity classifier and that of the coreference classifier.

While this ILP approach and Ng's (2004) approach to improving the output of an anaphoricity classifier both result in increased coreference performance, they have complementary strengths and weaknesses. Specifically, Ng's approach can directly optimize the desired coreference evaluation metric, but by treating the coreference system as a black box during the optimization process, it does not exploit the potentially useful pairwise probabilities provided by the coreference classifier. On the other hand, the ILP approach does exploit such pairwise probabilities, but optimizes an objective function that does not necessarily have any correlation with the desired evaluation metric.

Our goals in this paper are two-fold. First, motivated in part by previous work, we propose a graph-cut-based approach to anaphoricity determination that combines the strengths of Ng's approach and the ILP approach, by exploiting pairwise coreference probabilities when co-ordinating anaphoricity and coreference decisions, and at the same time allowing direct optimization of the desired coreference evaluation metric. Second, we compare our cut-based approach with the five aforementioned approaches to anaphoricity determination (namely, Ng and Cardie (2002a), Ng (2004), Luo (2007), Denis and Baldridge (2007), and Finkel and Manning (2008)) in terms of their effectiveness in improving a learning-based coreference system. To our knowledge, there has been no attempt to perform a comparative evaluation of existing approaches to anaphoricity determination. It is worth noting, in particular, that Luo (2007), Denis and Baldridge (2007), and Finkel and Manning (2008) evaluate their approaches on *true* mentions extracted from the answer keys. Since true mentions are composed of all the NPs involved in coreference relations but only a subset of the singleton NPs (i.e., NPs that are not coreferent with any other NPs) in a text, evaluating the utility of anaphoricity determination on true mentions to some extent defeats the purpose of performing anaphoricity determination, which precisely aims to identify non-anaphoric mentions. Hence, we hope that our evaluation on mentions extracted using an NP chunker can reveal their comparative strengths and weaknesses.

We perform our evaluation on three ACE coreference data sets using two commonly-used scoring programs. Experimental results show that (1) employing our cut-based approach to anaphoricity determination yields a coreference system that achieves the best performance for all six data-set/scoring-program combinations, and (2) among the five existing approaches, none performs consistently better than the others.

The rest of the paper is organized as follows. Section 2 describes our learning-based coreference system. In Section 3, we give an overview of the five baseline approaches to anaphoricity determination. Section 4 provides the details of our graph-cut-based approach. Finally, we present evaluation results in Section 5 and conclude in Section 6.

2 Baseline Coreference Resolution System

Our baseline coreference system implements the standard machine learning approach to coreference resolution (see Ng and Cardie (2002b), Ponzetto and Strube (2006), Yang and Su (2007), for instance), which consists of *probabilistic classification* and *clustering*, as described below.

2.1 The Standard Machine Learning Approach

We use maximum entropy (MaxEnt) classification (Berger et al., 1996) in conjunction with the 33 features described in Ng (2007) to acquire a model, P_C, for determining the probability that two mentions, m_i and m_j, are coreferent. Hence,

$$P_C(m_i, m_j) = P(\text{COREFERENT} \mid m_i, m_j).$$

In the rest of the paper, we will refer to $P_C(m_i, m_j)$ as the *pairwise coreference probability* between m_i and m_j. To generate training instances, we employ Soon et al.'s (2001) procedure, relying on the training texts to create (1) a *positive instance* for

each anaphoric mention, m_j, and its closest antecedent, m_i; and (2) a *negative instance* for m_j paired with each of the intervening mentions, m_{i+1}, m_{i+2},..., m_{j-1}. When training the feature-weight parameters of the MaxEnt model, we use 100 iterations of the improved iterative scaling (IIS) algorithm (Della Pietra et al., 1997) together with a Gaussian prior to prevent overfitting.

After training, the coreference model is used to select an antecedent for each mention in a test text. Following Soon et al. (2001), we select as the antecedent of each mention, m_j, the *closest* preceding mention that is classified as coreferent with m_j, where mention pairs with pairwise probabilities of at least 0.5 are considered coreferent. If no such mention exists, no antecedent will be selected for m_j. In essence, we use a *closest-first* clustering algorithm to impose a partitioning on the mentions.

3 Baseline Approaches to Anaphoricity Determination

As mentioned previously, we will use five existing approaches to anaphoricity determination as baselines in our evaluation. Common to all five approaches is the acquisition of an anaphoricity model, P_A, for determining the probability that a mention, m_j, is anaphoric. Hence,

$$P_A(m_j) = P(\text{ANAPHORIC} \mid m_j)$$

To train P_A, we again employ MaxEnt modeling, and create one training instance from each mention in a training text. Hence, each instance represents a single mention and consists of 37 features that are potentially useful for distinguishing anaphoric and non-anaphoric mentions (see Ng and Cardie (2002a) for a detailed description of these features).[1]

The classification of a training instance — one of ANAPHORIC or NOT ANAPHORIC — is derived directly from the coreference chains in the associated training text. Like the coreference model, the anaphoricity model is trained by running 100 iterations of IIS with a Guassian prior. The resulting model is then applied to a test text to determine the probability that a mention is anaphoric.

In the rest of this section, we provide an overview of the five baseline approaches to anaphoricity determination. We will characterize each approach along two dimensions: (1) whether it attempts to improve P_A, and if so, how; and (2) whether the resulting anaphoricity information is used as hard constraints or soft constraints by the coreference system.

3.1 Ng and Cardie (2002a)

Ng and Cardie (N&C) do not attempt to improve P_A, simply using the anaphoricity information it provides as hard constraints for coreference resolution. Specifically, the coreference system resolves only those mentions that are determined as anaphoric by P_A, where a mention is classified as anaphoric if the classification threshold is at least 0.5.

3.2 Ng (2004)

P_A may not be "sufficiently" accurate, however, as N&C report a significant drop in the performance of their coreference system after incorporating anaphoricity information, owing in part to their overly *conservative* anaphoricity model that misclassifies many anaphoric mentions as non-anaphoric. To address this problem, Ng (2004) attempts to improve P_A by introducing a threshold parameter c to adjust the conservativeness of P_A as follows. Given a specific c ($0 \leq c \leq 1$), a mention m_j is classified as anaphoric by P_A if and only if $P_A(m_j) \geq c$. It should be easy to see that decreasing c yields progressively less conservative anaphoricity models (i.e., more mentions will be classified as anaphoric). The parameter c is tuned using held-out development data to optimize the performance of the coreference system that employs anaphoricity information (as hard constraints).

In essence, Ng's approach to improving P_A treats the coreference system as a black box, merely selecting the value for c that yields the best score according to the desired coreference evaluation metric on the held-out data. In particular, unlike some of the anaphoricity determination approaches discussed later on, this approach does not attempt to coordinate the anaphoricity decisions and the pairwise coreference decisions. Nevertheless, as mentioned before, a unique strength of this approach lies in its ability to optimize directly the desired coreference

[1] While we train the anaphoricity model using the Ng and Cardie (2002a) feature set, it should be clear that any features that are useful for distinguishing anaphoric and non-anaphoric mentions can be used (e.g., those proposed by Uryupina (2003) and Elsner and Charniak (2007)).

evaluation metric.

3.3 Luo (2007)

Among the five anaphoricity determination approaches, Luo's (2007) is the only one where anaphoricity information is exploited as soft constraints by the coreference model, P_C.

Specifically, Luo's algorithm attempts to find the most probable coreference partition of a given set of mentions. To do so, it scores a partition using the probabilities provided by P_A and P_C. Let us illustrate how this can be done via the following example. Given a document with four mentions, $m_1, \ldots, m_4$, and a partition of the mentions, $\{[m_1, m_3, m_4], [m_2]\}$, automatically produced by some coreference system, Luo's algorithm scores the partition by considering the mentions in the document in a left-to-right manner. As the first mention in the document, m_1 is not anaphoric, and the probability that it is non-anaphoric is 1 – $P_A(m_1)$. Then, the algorithm processes m_2, which according to the partition is non-anaphoric, and the probability of its being non-anaphoric is 1 – $P_A(m_2)$. Next, it processes m_3, which is coreferent with m_1 with probability $P_C(m_1, m_3)$. Finally, it processes m_4, which is coreferent with m_1 and m_3. The probability that m_4 is coreferent with the cluster consisting of m_1 and m_3 is defined to be $\max(P_C(m_1, m_4), P_C(m_3, m_4))$, according to Luo's algorithm. The score of this partition is the product of these four probabilities, two provided by P_A and two by P_C. As can be seen, a partition is penalized whenever a mention that is unlikely to be anaphoric (according to P_A) is being resolved to some antecedent according to the partition.

Nevertheless, it is computationally infeasible to score all possible partitions given a set of mentions, as the number of partitions is exponential in the number of mentions. To cope with this computational complexity, Luo employs the algorithm proposed in Luo et al. (2004) to heuristically search for the most probable partition by performing a beam search through a *Bell tree*. In essence, only the most promising nodes in the tree are expanded at each step of the search process, where the "promise" of a node is defined in terms of the probabilities provided by P_A and P_C, as described above. Details of this process can be found in Luo et al. (2004).

3.4 Denis and Baldridge (2007)

As mentioned before, Denis and Baldridge (D&B) aim to improve the outputs of P_A and P_C by employing Integer Linear Programming (ILP) to perform joint inference for anaphoricity determination and coreference resolution. The ILP approach is motivated by the observation that the outputs of these two models have to satisfy certain constraints. For instance, if P_C determines that a mention, m_j, is not coreferent with any other mentions in the associated text, then P_A should determine that m_j is non-anaphoric. In practice, however, since P_A and P_C are trained independently of each other, this and other constraints cannot be enforced.

ILP provides a framework for *jointly* determining anaphoricity and coreference decisions for a given set of mentions based on the probabilities provided by P_A and P_C, such that the resulting joint decisions satisfy the desired constraints while respecting as much as possible the probabilistic decisions made by the independently-trained P_A and P_C. Specifically, an ILP program is composed of an objective function to be optimized subject to a set of linear constraints, and is created for each test text D as follows. Let M be the set of mentions in D, and P be the set of mention pairs formed from M (i.e., $P = \{(m_i, m_j) \mid m_i, m_j \in M, i < j\}$). Each ILP program has a set of indicator *variables*. In our case, we have one binary-valued variable for each anaphoricity decision and coreference decision to be made by an ILP solver. Following D&B's notation, we use y_j to denote the anaphoricity decision for mention m_j, and $x_{\langle i,j \rangle}$ to denote the coreference decision involving mentions m_i and m_j. In addition, each variable is associated with an *assignment* cost. Specifically, let $c^C_{\langle i,j \rangle} = -\log(P_C(m_i, m_j))$ be the cost of setting $x_{\langle i,j \rangle}$ to 1, and $\bar{c}^C_{\langle i,j \rangle} = -\log(1 - P_C(m_i, m_j))$ be the complementary cost of setting $x_{\langle i,j \rangle}$ to 0. We can similarly define the cost associated with each y_j, letting $c^A_j = -\log(P_A(m_j))$ be the cost of setting y_j to 1, and $\bar{c}^A_j = -\log(1 - P_A(m_j))$ be the complementary cost of setting y_j to 0. Given these costs, we aim to optimize the following objective function:

$$\min \sum_{(m_i,m_j) \in P} c^C_{\langle i,j \rangle} \cdot x_{\langle i,j \rangle} + \bar{c}^C_{\langle i,j \rangle} \cdot (1 - x_{\langle i,j \rangle}) + \sum_{m_j \in M} c^A_j \cdot y_j + \bar{c}^A_j \cdot (1 - y_j)$$

subject to a set of manually-specified *linear* constraints. D&B specify four types of constraints: (1) each indicator variable can take on a value of 0 or 1; (2) if m_i and m_j are coreferent ($x_{\langle i,j \rangle}$=1), then m_j is anaphoric (y_j=1); (3) if m_j is anaphoric (y_j=1), then it must be coreferent with some preceding mention m_i; and (4) if m_j is non-anaphoric, then it cannot be coreferent with any mention. Note that we are *minimizing* the objective function, since each assignment cost is expressed as a negative logarithm value. We use *lp_solve*[2], an ILP solver, to solve this program.

It is easy to see that enforcing consistency using ILP amounts to employing anaphoricity information as hard constraints for the coreference system. Since transitivity is not guaranteed by the above constraints, we follow D&B and use the *aggressive-merge* clustering algorithm to put any two mentions that are posited as coreferent into the same cluster.

3.5 Finkel and Manning (2008)

Finkel and Manning (F&M) present one simple extension to D&B's ILP approach: augmenting the set of linear constraints with the transitivity constraint. This ensures that if $x_{\langle i,j \rangle}$=1 and $x_{\langle j,k \rangle}$=1, then $x_{\langle i,k \rangle}$=1. As a result, the coreference decisions do not need to be co-ordinated by a separate clustering mechanism.

4 Cut-Based Anaphoricity Determination

As mentioned in the introduction, our graph-cut-based approach to anaphoricity determination is motivated by Ng's (2004) and the ILP approach, aiming to combine the strengths of the two approaches. Specifically, like Ng (2004), our approach allows direct optimization of the desired coreference evaluation metric; and like the ILP approach, our approach co-ordinates anaphoricity decisions and coreference decisions by exploiting the pairwise probabilities provided by a coreference model. In this section, we will introduce our cut-based approach, starting by reviewing concepts related to minimum cuts.

4.1 The Minimum Cut Problem Setting

Assume that we want to partition a set of n objects, $\{x_1, x_2, \ldots, x_n\}$, into two sets, Y_1 and Y_2. We have two types of scores concerning the x's and the Y's: *membership* scores and *similarity* scores. The membership score, $mem_{Y_i}(x_j)$, is a non-negative quantity that approximates the "affinity" of x_j to Y_i. On the other hand, the similarity score, $sim(x_j, x_k)$, is a non-negative quantity that provides an estimate of the similarity between x_j and x_k.

Informally, our goal is to maximize each object's net happiness, which is computed by subtracting its membership score of the class it is *not* assigned to from its membership score of the class it is assigned to. However, at the same time, we want to avoid assigning similar objects to different classes. More formally, we seek to minimize the partition cost:

$$\sum_{x_j \in Y_1, x_k \in Y_2} sim(x_j, x_k) + \sum_{x \in Y_1} mem_{Y_2}(x) + \sum_{x \in Y_2} mem_{Y_1}(x)$$

There exists an efficient algorithm for solving this seemingly intractable problem when it is recast as a graph problem. So, let us construct a graph, G, based on the available scores as follows. First, we create two nodes, s and t (called the *source* and the *sink*, respectively), to represent the two classes. Then, we create one "object" node for each of the n objects. For each object, x_j, we add two directed edges, one from s to x_j (with weight $mem_{Y_1}(x_j)$) and the other from x_j to t (with weight $mem_{Y_2}(x_j)$). Moreover, for each pair of object nodes, x_j and x_k, we add two directed edges (one from x_j to x_k and another from x_k to x_j), both of which have weight $sim(x_j, x_k)$. A *cut* in G is defined as a partition of the nodes into two sets, S and T, such that $s \in S$, $t \in T$; and the cost of the cut, $cost(S, T)$, is the sum of the weights of the edges going from S to T. A minimum cut is a cut that has the lowest cost among all the cuts of G. It can be proved that finding a minimum cut of G is equivalent to minimizing the partition cost defined as above. The main advantage of recasting the above minimization problem as a graph-cut problem is that there exist polynomial-time maxflow algorithms for finding a minimum cut.

4.2 Graph Construction

Next, we show how to construct the graph to which the mincut-finding algorithm will be applied. The ultimate goal is to use the mincut finder to partition a given set of mentions into two subsets, so that our coreference system will attempt to resolve only those mentions that are in the subset corresponding to ANAPHORIC. In other words, the resulting

[2] Available from http://lpsolve.sourceforge.net/

anaphoricity information will be used to identify and filter non-anaphoric mentions prior to coreference resolution. The graph construction process, which takes as input a set of mentions in a test text, is composed of three steps, as described below.

Step 1: Mimicking Ng and Cardie (2002a)

To construct the desired graph, G, we first create the source, s, and the sink, t, that represent the classes ANAPHORIC and NOT ANAPHORIC, respectively. Then, for each mention m_n in the input text, we create one node, n, and two edges, sn and nt, connecting n to s and t. Next, we compute w_{sn} and w_{nt}, the weights associated with sn and nt. A natural choice would be to use $P_A(m_n)$ as the weight of sn and $(1 - w_{sn})$ as the weight of nt. (We will assume throughout that w_{nt} is always equal to $1 - w_{sn}$.) If we apply the mincut finder to the current G, it should be easy to see that (1) any node n where $w_{sn} > 0.5$ will be assigned to s, (2) any node n where $w_{sn} < 0.5$ will be assigned to t, and (3) any remaining node will be assigned to one of them. (Without loss of generality, we assume that such nodes are assigned to s.) Hence, the set of mentions determined as anaphoric by the mincut finder is identical to the set of mentions classified as anaphoric by P_A, thus yielding a coreference system that is functionally equivalent to N&C's. This also implies that G shares the same potential weakness as P_A: being overly conservative in determining a mention as anaphoric.

Step 2: Mimicking Ng (2004)

One way to "improve" G is to make it functionally equivalent to Ng's (2004) approach. Specifically, our goal in Step 2 is to modify the edge weights in G (without adding new edges or nodes) such that the mincut finder classifies a node n as anaphoric if and only if $P_A(m_n) \geq c$ for some $c \in [0, 1]$. Now, recall from Step 1 that the mincut finder classifies a node n as anaphoric if and only if $w_{sn} \geq 0.5$. Hence, to achieve the aforementioned goal, we just need to ensure the property that $w_{sn} \geq 0.5$ if and only if $P_A(m_n) \geq c$. Consequently, we compute w_{sn} using a sigmoid function:

$$w_{sn} = \frac{1}{1 + e^{-\alpha \times (P_A(m_n) - c)}}$$

where α is a constant that controls the "steepness" of the sigmoid.[3] It should be easy to verify that the sigmoid satisfies the aforementioned property. As noted before, $w_{nt} = 1 - w_{sn}$ for each node n. Inspired by Ng (2004), the value of the parameter c will be tuned based on held-out development data to maximize coreference performance.

Step 3: Incorporating coreference probabilities

Like Ng's (2004) approach, the current G suffers from the weakness of not exploiting the pairwise probabilities provided by P_C. Fortunately, these probabilities can be naturally incorporated into G as similarity scores. To see why these pairwise probabilities are potentially useful, consider two mentions, m_i and m_j, in a text D that are coreferent and are both anaphoric. Assume that the graph G constructed from D has these edge weights: $w_{si} = 0.8$, $w_{sj} = 0.3$, and $w_{ij} = w_{ji} = 0.8$. Without the similarity scores, the mincut finder will correctly determine m_i as anaphoric but incorrectly classify m_j as non-anaphoric. On the other hand, if the similarity scores are taken into account, the mincut finder will correctly determine both mentions as anaphoric.

The above discussion suggests that it is desirable to incorporate edges between two nodes, i and j, when m_i and m_j are likely to be coreferent (i.e., $P_C(m_i, m_j) \geq c_2$ for some constant c_2). In our implementation, we tune this new parameter, c_2, jointly with c (see Step 2) on development data to maximize coreference performance. While it is possible to imagine scenarios where incorporating pairwise probabilities is not beneficial, we believe that these probabilities represent a source of information that could be profitably exploited via learning appropriate values for c and c_2.[4]

[3] One of the main reasons why we use a sigmoid function (rather than a linear function) is that the weights will still fall within the [0, 1] interval after the transformation, a property that will turn out to be convenient when the pairwise coreference probabilities are incorporated (see Step 3). α is chosen so that the difference between two weights after the transformation is as close as possible to their difference before the transformation. With this criterion in mind, we set α to 0.42 in our experiments.

[4] Incorporating the coreference probabilities can potentially identify some of the anaphoric mentions that would be misclassified otherwise. However, note that the minimum cut algorithm does not maintain the notion of directionality that would allow one to determine that a discourse-new mention (i.e., the first mention of a coreference chain) is not anaphoric. In particular, the algorithm tends to classify all members of a coreference chain, including the first mention, as anaphoric. We did not ex-

5 Evaluation

5.1 Experimental Setup

For evaluation, we use the ACE Phase II coreference corpus, which is composed of three sections: Broadcast News (BNEWS), Newspaper (NPAPER), and Newswire (NWIRE). Each section is in turn composed of a training set and a test set. For each section, we train an anaphoricity model, P_A, and a coreference model, P_C, on the training set, and evaluate P_C (when used in combination with different approaches to anaphoricity determination) on the test set. As noted before, the mentions used are extracted automatically using an in-house NP chunker. Results are reported in terms of recall (R), precision (P), and F-measure (F), obtained using two coreference scoring programs: the MUC scorer (Vilain et al., 1995) and the CEAF scorer (Luo, 2005).

5.2 Results and Discussions

"No Anaphoricity" baseline. Our first baseline is the learning-based coreference system described in Section 2, which does not employ any anaphoricity determination algorithm. Results using the MUC scorer and the CEAF scorer are shown in row 1 of Tables 1 and 2, respectively. As we can see, MUC F-score ranges from 55.0 to 61.7 and CEAF F-score ranges from 55.3 to 61.2.

Duplicated Ng and Cardie (2002a) baseline. Next, we evaluate our second baseline, which is N&C's coreference system. As seen from row 2 of Tables 1 and 2, MUC F-score ranges from 50.5 to 60.0 and CEAF F-score ranges from 54.5 to 59.4. In comparison to the first baseline, we see drops in F-score in all cases as a result of considerable precipitation in recall, which can in turn be attributed to the misclassification of many anaphoric mentions by the anaphoricity model. More specifically, MUC F-score decreases by 1.7–5.5%, whereas CEAF F-score decreases by 0.5–1.8%. These trends are consistent with those reported in N&C's paper.

Duplicated Ng (2004) baseline. Our third baseline is Ng's (2004) coreference system. Recall that this resolver requires the tuning of the conservativeness parameter, c, on held-out data. To ensure a fair comparison between different resolvers, we do not rely on additional data for parameter tuning. Rather, we reserve $\frac{1}{3}$ of the available training data for tuning c, for which we tested values from 0 to 1 in steps of 0.01, and use the remaining $\frac{2}{3}$ of the data for training P_A and P_C. Results are shown in row 3 of Tables 1 and 2, where MUC F-score ranges from 57.0 to 61.9 and CEAF F-score ranges from 55.5 to 60.6. In comparison to the first baseline, we obtain mixed results: MUC F-score increases by 2.0% and 0.2% for BNEWS and NPAPER, respectively, but drops by 0.1% for NWIRE; CEAF F-score increases by 0.2% and 1.1% for BNEWS and NPAPER, respectively, but drops by 0.6% for NWIRE.

Duplicated Luo (2007) baseline. Results of our fourth baseline, in which the anaphoricity and pairwise coreference probabilities are combined to score a partition using Luo's system, are shown in row 4 of Tables 1 and 2. Here, we see that MUC F-score ranges from 55.8 to 62.1 and CEAF F-score ranges from 56.3 to 61.5. In comparison to the first baseline, performance improves, though insignificantly,[5] in all cases: MUC F-score increases by 0.2–0.8%, whereas CEAF F-score increases by 0.3–1.0%.

Duplicated Denis and Baldridge (2007) baseline. Our fifth baseline performs joint inference for anaphoricity determination and coreference resolution using D&B's ILP approach. Results are shown in row 5 of Tables 1 and 2, where MUC F-score ranges from 56.2 to 63.8 and CEAF F-score ranges from 56.9 to 61.5. In comparison to the first baseline, MUC F-score always increases, with improvements ranging from 1.2% to 2.1%. CEAF results are mixed: F-score increases significantly for BNEWS, drops insignificantly for NPAPER, and rises insignificantly for NWIRE. The difference in performance trends between the two scorers can be attributed to the fact that the MUC scorer typically under-penalizes errors due to overmerging, which occurs as a result of D&B's using the aggressive-merge clustering algorithm. In addition, we can see that D&B's approach performs at least as good as Luo's approach in all but one case (NPAPER/CEAF).

Duplicated Finkel and Manning (2008) baseline. Our sixth baseline is F&M's coreference system,

plicitly address this issue, simply letting the coreference clustering algorithm discover that first mentions are non-anaphoric.

[5] Like the MUC organizers, we use Approximate Randomization (Noreen, 1989) for significance testing, with p=0.05.

	Approach to Anaphoricity Determination	Broadcast News R	Broadcast News P	Broadcast News F	Newspaper R	Newspaper P	Newspaper F	Newswire R	Newswire P	Newswire F
1	No Anaphoricity	57.7	52.6	55.0	60.8	62.6	61.7	59.1	58.1	58.6
2	Duplicated Ng and Cardie (2002a)	40.3	67.7	50.5†	52.1	70.6	60.0	43.0	69.3	53.1†
3	Duplicated Ng (2004)	51.9	63.2	57.0	60.0	63.8	61.9	59.3	57.7	58.5
4	Duplicated Luo (2007)	55.4	56.1	55.8	60.6	63.7	62.1	58.4	59.2	58.8
5	Duplicated Denis and Baldridge (2007)	57.3	55.1	56.2*	63.8	63.7	63.8*	60.4	59.3	59.8*
6	Duplicated Finkel and Manning (2008)	56.4	55.3	55.8	63.8	63.7	63.8*	59.7	59.2	59.5
7	Graph Minimum Cut	53.1	67.5	**59.4***	57.9	71.2	**63.9***	54.1	69.0	**60.6***

Table 1: MUC scores for the three ACE data sets. F-scores that represent statistically significant gains and drops with respect to the "No Anaphoricity" baseline are marked with an asterisk (*) and a dagger (†), respectively.

	Approach to Anaphoricity Determination	Broadcast News R	Broadcast News P	Broadcast News F	Newspaper R	Newspaper P	Newspaper F	Newswire R	Newswire P	Newswire F
1	No Anaphoricity	63.2	49.2	55.3	64.5	54.3	59.0	67.3	56.1	61.2
2	Duplicated Ng and Cardie (2002a)	55.9	53.3	54.5	60.7	56.3	58.5	60.6	58.2	59.4
3	Duplicated Ng (2004)	62.5	49.9	55.5	63.5	57.0	60.1	65.6	56.3	60.6
4	Duplicated Luo (2007)	62.7	51.1	56.3	64.6	55.4	59.6	67.0	56.8	61.5
5	Duplicated Denis and Baldridge (2007)	63.8	51.4	56.9*	62.6	53.6	57.8	67.0	56.8	61.5
6	Duplicated Finkel and Manning (2008)	63.2	51.3	56.7*	62.6	53.6	57.8	66.7	56.7	61.3
7	Graph Minimum Cut	61.4	57.6	**59.4***	64.1	59.4	**61.7***	65.7	61.9	**63.8***

Table 2: CEAF scores for the three ACE data sets. F-scores that represent statistically significant gains and drops with respect to the "No Anaphoricity" baseline are marked with an asterisk (*) and a dagger (†), respectively.

which is essentially D&B's approach augmented with transitivity constraints. Results are shown in row 6 of Tables 1 and 2, where MUC F-score ranges from 55.8 to 63.8 and CEAF F-score ranges from 56.7 to 61.3. In comparison to the D&B baseline, we see that F-score never improves, regardless of which scoring program is used. In fact, recall slightly deteriorates, and this can be attributed to F&M's observation that transitivity constraints tend to produce smaller clusters. Overall, these results suggest that enforcing transitivity for coreference resolution is not useful for improving coreference performance.

Our graph-cut-based approach. Finally, we evaluate the coreference system using the anaphoricity information provided by our cut-based approach. As before, we reserve $\frac{1}{3}$ of the training data for *jointly* tuning the two parameters, c and c_2, and use the remaining $\frac{2}{3}$ for training P_A and P_C. For tuning, we tested values from 0 to 1 in steps of 0.1 for both c and c_2. Results are shown in row 7 of Tables 1 and 2. As we can see, MUC F-score ranges from 59.4 to 63.9 and CEAF F-score ranges from 59.4 to 63.8, representing a significant improvement over the first baseline in all six cases: MUC F-score rises by 2.0–4.4% and CEAF F-score rises by 2.6–4.1%. Such an improvement can be attributed to a large gain in precision and a smaller drop in recall. This implies that our mincut algorithm has successfully identified many non-anaphoric mentions, but in comparison to N&C's approach, it misclassifies a smaller number of anaphoric mentions. Moreover, our approach achieves the best F-score for each data-set/scoring-program combination, and significantly outperforms the best baseline (D&B) in all but two cases, NPAPER/MUC and NWIRE/MUC.

6 Conclusions

We have presented a graph-cut-based approach to anaphoricity determination that (1) directly optimizes the desired coreference evaluation metric through parameterization and (2) exploits the probabilities provided by the coreference model when coordinating anaphoricity and coreference decisions. Another major contribution of our work is the empirical comparison of our approach against five existing approaches to anaphoricity determination in terms of their effectiveness in improving a coreference system using automatically extracted mentions. Our approach demonstrates effectiveness and robustness by achieving the best result on all three ACE data sets according to both the MUC scorer and the CEAF scorer. We believe that our cut-based approach provides a flexible mechanism for coordinating anaphoricity and coreference decisions.

Acknowledgments

We thank the three anonymous reviewers for their invaluable comments, Kazi Saidul Hasan for his help on using *lp_solve*, and NSF for its gracious support of this work under Grant IIS-0812261. The description of the minimum cut framework in Section 4.1 was inspired by Pang and Lee (2004).

References

D. Bean and E. Riloff. 1999. Corpus-based identification of non-anaphoric noun phrases. In *Proc. of the ACL*, pages 373–380.

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

S. Bergsma, D. Lin, and R. Goebel. 2008. Distributional identification of non-referential pronouns. In *Proc. of ACL-08:HLT*, pages 10–18.

S. Della Pietra, V. Della Pietra, and J. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

P. Denis and J. Baldridge. 2007. Global, joint determination of anaphoricity and coreference resolution using integer programming. In *Proc. of NAACL/HLT*, pages 236–243.

M. Elsner and E. Charniak. 2007. A generative discourse-new model for text coherence. *Technical Report CS-07-04*, Brown University.

R. Evans. 2001. Applying machine learning toward an automatic classification of *it*. *Literary and Linguistic Computing*, 16(1):45–57.

J. R. Finkel and C. Manning. 2008. Enforcing transitivity in coreference resolution. In *Proc. of ACL-08:HLT Short Papers (Companion Volume)*, pages 45–48.

C. Kennedy and B. Boguraev. 1996. Anaphor for everyone: Pronominal anaphora resolution without a parser. In *Proc. of COLING*, pages 113–118.

S. Lappin and H. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–562.

X. Luo, A. Ittycheriah, H. Jing, N. Kambhatla, and S. Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *Proc. of the ACL*, pages 135–142.

X. Luo. 2005. On coreference resolution performance metrics. In *Proc. of HLT/EMNLP*, pages 25–32.

X. Luo. 2007. Coreference or not: A twin model for coreference resolution. In *Proc. of NAACL-HLT*, pages 73–80.

C. Müller. 2006. Automatic detection of nonreferential it in spoken multi-party dialog. In *Proc. of EACL*, pages 49–56.

V. Ng. 2007. Shallow semantics for coreference resolution. In *Proceedings of IJCAI*, pages 1689–1694.

V. Ng and C. Cardie. 2002a. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proc. of COLING*, pages 730–736.

V. Ng and C. Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proc. of the ACL*, pages 104–111.

V. Ng. 2004. Learning noun phrase anaphoricity to improve conference resolution: Issues in representation and optimization. In *Proc. of the ACL*, pages 151–158.

E. W. Noreen. 1989. *Computer Intensive Methods for Testing Hypothesis: An Introduction*. John Wiley & Sons.

C. Paice and G. Husk. 1987. Towards the automatic recognition of anaphoric features in English text: the impersonal pronoun 'it'. *Computer Speech and Language*, 2.

B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. of the ACL*, pages 271–278.

M. Poesio, O. Uryupina, R. Vieira, M. Alexandrov-Kabadjov, and R. Goulart. 2004. Discourse-new detectors for definite description resolution: A survey and a preliminary proposal. In *Proc. of the ACL Workshop on Reference Resolution*.

S. P. Ponzetto and M. Strube. 2006. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proc. of HLT/NAACL*, pages 192–199.

W. M. Soon, H. T. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

O. Uryupina. 2003. High-precision identification of discourse new and unique noun phrases. In *Proc. of the ACL Student Research Workshop*.

Y. Versley, A. Moschitti, M. Poesio, and X. Yang. 2008. Coreference systems based on kernels methods. In *Proc. of COLING*, pages 961–968.

R. Vieira and M. Poesio. 2000. Processing definite descriptions in corpora. In S. Botley and A. McEnery, editors, *Corpus-based and Computational Approaches to Discourse Anaphora*, pages 189–212. UCL Press.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proc. of MUC-6*, pages 45–52.

B. L. Webber. 1979. *A Formal Approach to Discourse Anaphora*. Garland Publishing, Inc.

X. Yang and J. Su. 2007. Coreference resolution using semantic relatedness information from automatically discovered patterns. In *Proc. of ACL*, pages 528–535.

Using Citations to Generate Surveys of Scientific Paradigms

Saif Mohammad[†*], Bonnie Dorr[†‡*], Melissa Egan[†‡], Ahmed Hassan[ϕ], Pradeep Muthukrishan[ϕ], Vahed Qazvinian[ϕ], Dragomir Radev[ϕ§], David Zajic[‡◇]

Institute for Advanced Computer Studies[†] and Computer Science[‡], University of Maryland.
Human Language Technology Center of Excellence[*]. Center for Advanced Study of Language[◇].
{saif,bonnie,mkegan,dmzajic}@umiacs.umd.edu

Department of Electrical Engineering and Computer Science[ϕ]
School of Information[§], University of Michigan.
{hassanam,mpradeep,vahed,radev}@umich.edu

Abstract

The number of research publications in various disciplines is growing exponentially. Researchers and scientists are increasingly finding themselves in the position of having to quickly understand large amounts of technical material. In this paper we present the first steps in producing an automatically generated, readily consumable, technical survey. Specifically we explore the combination of citation information and summarization techniques. Even though prior work (Teufel et al., 2006) argues that citation text is unsuitable for summarization, we show that in the framework of multi-document survey creation, citation texts can play a crucial role.

1 Introduction

In today's rapidly expanding disciplines, scientists and scholars are constantly faced with the daunting task of keeping up with knowledge in their field. In addition, the increasingly interconnected nature of real-world tasks often requires experts in one discipline to rapidly learn about other areas in a short amount of time.

Cross-disciplinary research requires scientists in areas such as linguistics, biology, and sociology to learn about computational approaches and applications, e.g., computational linguistics, biological modeling, social networks. Authors of journal articles and books must write accurate surveys of previous work, ranging from short summaries of related research to in-depth historical notes.

Interdisciplinary review panels are often called upon to review proposals in a wide range of areas, some of which may be unfamiliar to panelists. Thus, they must learn about a new discipline "on the fly" in order to relate their own expertise to the proposal.

Our goal is to effectively serve these needs by combining two currently available technologies: (1) bibliometric lexical link mining that exploits the structure of citations and relations among citations; and (2) summarization techniques that exploit the content of the material in both the citing and cited papers.

It is generally agreed upon that manually written abstracts are good summaries of individual papers. More recently, Qazvinian and Radev (2008) argue that **citation texts** are useful in creating a summary of the important contributions of a research paper. The citation text of a target paper is the set of sentences in other technical papers that explicitly refer to it (Elkiss et al., 2008a). However, Teufel (2005) argues that using citation text directly is not suitable for document summarization.

In this paper, we compare and contrast the usefulness of abstracts and of citation text in automatically generating a technical survey on a given topic from multiple research papers. The next section provides the background for this work, including the primary features of a technical survey and also the types of input that are used in our study (full papers, abstracts, and citation texts). Following this, we describe related work and point out the advances of our work over previous work. We then describe how citation texts are used as a new input for multi-document summarization to produce surveys of a given technical area. We apply four different summarization techniques to data in the ACL Anthol-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 584–592,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

ogy and evaluate our results using both automatic (ROUGE) and human-mediated (nugget-based pyramid) measures. We observe that, as expected, abstracts are useful in survey creation, but, notably, we also conclude that citation texts have crucial survey-worthy information not present in (or at least, not easily extractable from) abstracts. We further discover that abstracts are author-biased and thus complementary to the broader perspective inherent in citation texts; these differences enable the use of a range of different levels and types of information in the survey—the extent of which is subject to survey length restrictions (if any).

2 Background

Automatically creating technical surveys is significantly distinct from that of traditional multi-document summarization. Below we describe primary characteristics of a technical survey and we present three types of input texts that we used for the production of surveys.

2.1 Technical Survey

In the case of multi-document summarization, the goal is to produce a readable presentation of multiple documents, whereas in the case of technical survey creation, the goal is to convey the key features of a particular field, basic underpinnings of the field, early and late developments, important contributions and findings, contradicting positions that may reverse trends or start new sub-fields, and basic definitions and examples that enable rapid understanding of a field by non-experts.

A prototypical example of a technical survey is that of "chapter notes," i.e., short (50–500 word) descriptions of sub-areas found at the end of chapters of textbook, such as Jurafsky and Martin (2008). One might imagine producing such descriptions automatically, then hand-editing them and refining them for use in an actual textbook.

We conducted a human analysis of these chapter notes that revealed a set of conventions, an outline of which is provided here (with example sentences in italics):

1. Introductory/opening statement: *The earliest computational use of X was in Y, considered by many to be the foundational work in this area.*
2. Definitional follow up: *X is defined as Y.*
3. Elaboration of definition (e.g., with an example): *Most early algorithms were based on Z.*
4. Deeper elaboration, e.g., pointing out issues with initial approaches: *Unfortunately, this model seems to be wrong.*
5. Contrasting definition: *Algorithms since then...*
6. Introduction of additional specific instances / historical background with citations: *Two classic approaches are described in Q.*
7. References to other summaries: *R provides a comprehensive guide to the details behind X.*

The notion of *text level categories* or *zoning* of technical papers—related to the survey components enumerated above—has been investigated previously in the work of Nanba and Kan (2004b) and Teufel (2002). These earlier works focused on the *analysis* of scientific papers based on their rhetorical structure and on determining the portions of papers that contain new results, comparisons to earlier work, etc. The work described in this paper focuses on the *synthesis* of technical surveys based on knowledge gleaned from rhetorical structure not unlike that of the work of these earlier researchers, but perhaps guided by structural patterns along the lines of the conventions listed above.

Although our current approach to survey creation does not yet incorporate a fully pattern-based component, our ultimate objective is to apply these patterns to guide the creation and refinement of the final output. As a first step toward this goal, we use citation texts (closest in structure to the patterns identified by convention 7 above) to pick out the most important content for survey creation.

2.2 Full papers, abstracts, and citation texts

Published research on a particular topic can be summarized from two different kinds of sources: (1) where an author describes her own work and (2) where others describe an author's work (usually in relation to their own work). The author's description of her own work can be found in her paper. How others perceive her work is spread across other papers that cite her work. We will refer to the set of sentences that explicitly mention a target paper Y as the citation text of Y.

Traditionally, technical survey generation has been tackled by summarizing a set of research papers pertaining to the topic. However, individual research papers usually come with manually-created "summaries"—their abstracts. The abstract of a paper may have sentences that set the context, state the problem statement, mention how the problem is approached, and the bottom-line results—all in 200 to 500 words. Thus, using only the abstracts (instead of full papers) as input to a summarization system is worth exploring.

Whereas the abstract of a paper presents what the authors think to be the important contributions of a paper, the citation text of a paper captures what others in the field perceive as the contributions of the paper. The two perspectives are expected to have some overlap in their content, but the citation text also contains additional information not found in abstracts (Elkiss et al., 2008a). For example, how a particular methodology (described in one paper) was combined with another (described in a different paper) to overcome some of the drawbacks of each. A citation text is also an indicator of what contributions described in a paper were more influential over time. Another distinguishing feature of citation texts in contrast to abstracts is that a citation text tends to have a certain amount of redundant information. This is because multiple papers may describe the same contributions of a target paper. This redundancy can be exploited to determine the important contributions of the target paper.

Our goal is to test the hypothesis that an effective technical survey will reflect information on research not only from the perspective of its authors but also from the perspective of others who use/commend/discredit/add to it. Before describing our experiments with technical papers, abstracts, and citation texts, we first summarize relevant prior work that used these sources of information as input.

3 Related work

Previous work has focused on the analysis of citation and collaboration networks (Teufel et al., 2006; Newman, 2001) and scientific article summarization (Teufel and Moens, 2002). Bradshaw (2003) used citation texts to determine the content of articles and improve the results of a search engine. Citation texts have also been used to create summaries of single scientific articles in Qazvinian and Radev (2008) and Mei and Zhai (2008). However, there is no previous work that uses the text of the citations to produce a multi-document survey of scientific articles. Furthermore, there is no study contrasting the quality of surveys generated from citation summaries—both automatically and manually produced—to surveys generated from other forms of input such as the abstracts or full texts of the source articles.

Nanba and Okumura (1999) discuss citation categorization to support a system for writing a survey. Nanba et al. (2004a) automatically categorize citation sentences into three groups using pre-defined phrase-based rules. Based on this categorization a survey generation tool is introduced in Nanba et al. (2004b). They report that co-citation (where both papers are cited by many other papers) implies similarity by showing that the textual similarity of co-cited papers is proportional to the proximity of their citations in the citing article.

Elkiss et al. (2008b) conducted several experiments on a set of 2,497 articles from the free PubMed Central (PMC) repository.[1] Results from this experiment confirmed that the cohesion of a citation text of an article is consistently higher than the that of its abstract. They also concluded that citation texts contain additional information are more focused than abstracts.

Nakov et al. (2004) use sentences surrounding citations to create training and testing data for semantic analysis, synonym set creation, database curation, document summarization, and information retrieval. Kan et al. (2002) use annotated bibliographies to cover certain aspects of summarization and suggest using metadata and critical document features as well as the prominent content-based features to summarize documents. Kupiec et al. (1995) use a statistical method and show how extracts can be used to create summaries but use no annotated metadata in summarization.

Siddharthan and Teufel (2007) describe a new reference task and show high human agreement as well as an improvement in the performance of **argumentative zoning** (Teufel, 2005). In argumentative zoning—a rhetorical classification task—seven

[1] http://www.pubmedcentral.gov

classes (Own, Other, Background, Textual, Aim, Basis, and Contrast) are used to label sentences according to their role in the author's argument.

Our aim is not only to determine the utility of citation texts for survey creation, but also to examine the quality distinctions between this form of input and others such as abstracts and full texts—comparing the results to human-generated surveys using both automatic and nugget-based pyramid evaluation (Lin and Demner-Fushman, 2006; Nenkova and Passonneau, 2004; Lin, 2004).

4 Summarization systems

We used four summarization systems for our survey-creation approach: **Trimmer, LexRank, C-LexRank,** and **C-RR**. Trimmer is a syntactically-motivated parse-and-trim approach. LexRank is a graph-based similarity approach. C-LexRank and C-RR use graph clustering ('C' stands for clustering). We describe each of these, in turn, below.

4.1 Trimmer

Trimmer is a sentence-compression tool that extends the scope of an extractive summarization system by generating multiple alternative sentence compressions of the most important sentences in target documents (Zajic et al., 2007). Trimmer compressions are generated by applying linguistically-motivated rules to mask syntactic components of a parse of a source sentence. The rules can be applied iteratively to compress sentences below a configurable length threshold, or can be applied in all combinations to generate the full space of compressions.

Trimmer can leverage the output of any constituency parser that uses the Penn Treebank conventions. At present, the Stanford Parser (Klein and Manning, 2003) is used. The set of compressions is ranked according to a set of features that may include metadata about the source sentences, details of the compression process that generated the compression, and externally calculated features of the compression.

Summaries are constructed from the highest scoring compressions, using the metadata and maximal marginal relevance (Carbonell and Goldstein, 1998) to avoid redundancy and over-representation of a single source.

4.2 LexRank

We also used LexRank (Erkan and Radev, 2004), a state-of-the-art multidocument summarization system, to generate summaries. LexRank first builds a graph of all the candidate sentences. Two candidate sentences are connected with an edge if the similarity between them is above a threshold. We used cosine as the similarity metric with a threshold of 0.15. Once the network is built, the system finds the most central sentences by performing a random walk on the graph.

The salience of a node is recursively defined on the salience of adjacent nodes. This is similar to the concept of prestige in social networks, where the prestige of a person is dependent on the prestige of the people he/she knows. However, since random walk may get caught in cycles or in disconnected components, we reserve a low probability to jump to random nodes instead of neighbors (a technique suggested by Langville and Meyer (2006)).

Note also that unlike the original PageRank method, the graph of sentences is undirected. This updated measure of sentence salience is called as LexRank. The sentences with the highest LexRank scores form the summary.

4.3 Cluster Summarizers: C-LexRank, C-RR

Two clustering methods proposed by Qazvinian and Radev (2008)—C-RR and C-LexRank—were used to create summaries. Both create a fully connected network in which nodes are sentences and edges are cosine similarities. A cutoff value of 0.1 is applied to prune the graph and make a binary network. The largest connected component of the network is then extracted and clustered.

Both of the mentioned summarizers cluster the network similarly but use different approaches to select sentences from different communities. In C-RR sentences are picked from different clusters in a round robin (RR) fashion. C-LexRank first calculates LexRank within each cluster to find the most salient sentences of each community. Then it picks the most salient sentence of each cluster, and then the second most salient and so forth until the summary length limit is reached.

Most of work in QA and paraphrasing focused on folding paraphrasing knowledge into question analyzer or answer locator Rinaldi et al, 2003; Tomuro, 2003. In addition, number of researchers have built systems to take reading comprehension examinations designed to evaluate children's reading levels Charniak et al, 2000; Hirschman et al, 1999; Ng et al, 2000; Riloff and Thelen, 2000; Wang et al, 2000. so-called " definition " or " other " questions at recent TREC evalua - tions Voorhees, 2005 serve as good examples. To better facilitate user information needs, recent trends in QA research have shifted towards complex, context-based, and interactive question answering Voorhees, 2001; Small et al, 2003; Harabagiu et al, 2005. [And so on.]

Table 1: First few sentences of the QA citation texts survey generated by Trimmer.

5 Data

The *ACL Anthology* is a collection of papers from the Computational Linguistics journal, and proceedings of ACL conferences and workshops. It has almost 11,000 papers. To produce the **ACL Anthology Network (AAN)**, Joseph and Radev (2007) manually parsed the references before automatically compiling the network metadata, and generating citation and author collaboration networks. The AAN includes all citation and collaboration data within the ACL papers, with the citation network consisting of 11,773 nodes and 38,765 directed edges.

Our evaluation experiments are on a set of papers in the research area of Question Answering (QA) and another set of papers on Dependency parsing (DP). The two sets of papers were compiled by selecting all the papers in AAN that had the words *Question Answering* and *D*ependency Parsing, respectively, in the title and the content. There were 10 papers in the QA set and 16 papers in the DP set. We also compiled the citation texts for the 10 QA papers and the citation texts for the 16 DP papers.

6 Experiments

We automatically generated surveys for both QA and DP from three different types of documents: (1) full papers from the QA and DP sets—**QA and DP full papers (PA)**, (2) only the abstracts of the QA and DP papers—**QA and DP abstracts (AB)**, and (3) the citation texts corresponding to the QA and DP papers—**QA and DP citations texts (CT)**.

We generated twenty four (4x3x2) surveys, each of length 250 words, by applying Trimmer, LexRank, C-LexRank and C-RR on the three data types (citation texts, abstracts, and full papers) for both QA and DP. (Table 1 shows a fragment of one of the surveys automatically generated from QA citation texts.) We created six (3x2) additional 250-word surveys by randomly choosing sentences from the citation texts, abstracts, and full papers of QA and DP. We will refer to them as **random surveys**.

6.1 Evaluation

Our goal was to determine if citation texts do indeed have useful information that one will want to put in a survey and if so, how much of this information is **not** available in the original papers and their abstracts. For this we evaluated each of the automatically generated surveys using two separate approaches: nugget-based pyramid evaluation and ROUGE (described in the two subsections below).

Two sets of gold standard data were manually created from the QA and DP citation texts and abstracts, respectively:[2] (1) We asked two impartial judges to identify important nuggets of information worth including in a survey. (2) We asked four fluent speakers of English to create 250-word surveys of the datasets. Then we determined how well the different automatically generated surveys perform against these gold standards. If the citation texts have only redundant information with respect to the abstracts and original papers, then the surveys of citation texts will not perform better than others.

6.1.1 Nugget-Based Pyramid Evaluation

For our first approach we used a nugget-based evaluation methodology (Lin and Demner-Fushman, 2006; Nenkova and Passonneau, 2004; Hildebrandt et al., 2004; Voorhees, 2003). We asked three impartial annotators (knowledgeable in NLP but not affiliated with the project) to review the citation texts and/or abstract sets for each of the papers in the QA and DP sets and manually extract prioritized lists

[2]Creating gold standard data from complete papers is fairly arduous, and was not pursued.

of 2–8 "nuggets," or main contributions, supplied by each paper. Each nugget was assigned a weight based on the frequency with which it was listed by annotators as well as the priority it was assigned in each case. Our automatically generated surveys were then scored based on the number and weight of the nuggets that they covered. This evaluation approach is similar to the one adopted by Qazvinian and Radev (2008), but adapted here for use in the multi-document case.

The annotators had two distinct tasks for the QA set, and one for the DP set: (1) extract nuggets for each of the 10 QA papers, based only on the citation texts for those papers; (2) extract nuggets for each of the 10 QA papers, based only on the abstracts of those papers; and (3) extract nuggets for each of the 16 DP papers, based only on the citation texts for those papers.[3]

We obtained a weight for each nugget by reversing its priority out of 8 (e.g., a nugget listed with priority 1 was assigned a weight of 8) and summing the weights over each listing of that nugget.[4]

To evaluate a given survey, we counted the number and weight of nuggets that it covered. Nuggets were detected via the combined use of annotator-provided regular expressions and careful human review. Recall was calculated by dividing the combined weight of covered nuggets by the combined weight of all nuggets in the nugget set. Precision was calculated by dividing the number of distinct nuggets covered in a survey by the number of sentences constituting that survey, with a cap of 1. F-measure, the weighted harmonic mean of precision and recall, was calculated with a beta value of 3 in order to assign the greatest weight to recall. Recall is favored because it rewards surveys that include highly weighted (important) facts, rather than just a great number of facts.

[3]We first experimented using only the QA set. Then to show that the results apply to other datasets, we asked human annotators for gold standard data on the DP citation texts. Additional experiments on DP abstracts were not pursued because this would have required additional human annotation effort to establish a point we had already made with the QA set, i.e., that abstracts are useful for survey creation.

[4]Results obtained with other weighting schemes that ignored priority ratings and multiple mentions of a nugget by a single annotator showed the same trends as the ones shown by the selected weighting scheme, but the latter was a stronger distinguisher among the four systems.

	Human Performance: Pyramid F-measure				
	Human1	Human2	Human3	Human4	Average
Input: QA citation surveys					
QA–CT nuggets	0.524	0.711	0.468	0.695	0.599
QA–AB nuggets	0.495	0.606	0.423	0.608	0.533
Input: QA abstract surveys					
QA–CT nuggets	0.542	0.675	0.581	0.669	0.617
QA–AB nuggets	0.646	0.841	0.673	0.790	0.738
Input: DP citation surveys					
DP–CT nuggets	0.245	0.475	0.378	0.555	0.413

Table 2: Pyramid F-measure scores of human-created surveys of QA and DP data. The surveys are evaluated using nuggets drawn from QA citation texts (QA–CT), QA abstracts (QA–AB), and DP citation texts (DP–CT).

Table 2 gives the F-measure values of the 250-word surveys manually generated by humans. The surveys were evaluated using the nuggets drawn from the QA citation texts, QA abstracts, and DP citation texts. The average of their scores (listed in the rightmost column) may be considered a good score to aim for by the automatic summarization methods.

Table 3 gives the F-measure values of the surveys generated by the four automatic summarizers, evaluated using nuggets drawn from the QA citation texts, QA abstracts, and DP citation texts. The table also includes results for the baseline random summaries.

When we used the nuggets from the abstracts set for evaluation, the surveys created from abstracts scored higher than the corresponding surveys created from citation texts and papers. Further, the best surveys generated from citation texts outscored the best surveys generated from papers. **When we used the nuggets from citation sets for evaluation,** the best automatic surveys generated from citation texts outperform those generated from abstracts and full papers. All these pyramid results demonstrate that citation texts can contain useful information that is not available in the abstracts or the original papers, and that abstracts can contain useful information that is not available in the citation texts or full papers.

Among the various automatic summarizers, Trimmer performed best at this task, in two cases exceeding the average human performance. Note also that the random summarizer outscored the automatic summarizers in cases where the nuggets were taken from a source different from that used to generate the survey. However, one or two summarizers still tended to do well. This indicates a difficulty in ex-

	System Performance: Pyramid F-measure				
	Random	C-LexRank	C-RR	LexRank	Trimmer
Input: QA citation surveys					
QA–CT nuggets	0.321	0.434	0.268	0.295	**0.616**
QA–AB nuggets	0.305	0.388	0.349	0.320	**0.543**
Input: QA abstract surveys					
QA–CT nuggets	0.452	0.383	**0.480**	0.441	0.404
QA–AB nuggets	**0.623**	0.484	0.574	0.606	0.622
Input: QA full paper surveys					
QA–CT nuggets	0.239	**0.446**	0.299	0.190	0.199
QA–AB nuggets	0.294	**0.520**	0.387	0.301	0.290
Input: DP citation surveys					
DP–CT nuggets	0.219	0.231	0.170	**0.372**	0.136
Input: DP abstract surveys					
DP–CT nuggets	**0.321**	0.301	0.263	0.311	0.312
Input: DP full paper surveys					
DP–CT nuggets	0.032	0.000	0.144	*	**0.280**

Table 3: Pyramid F-measure scores of automatic surveys of QA and DP data. The surveys are evaluated using nuggets drawn from QA citation texts (QA–CT), QA abstracts (QA–AB), and DP citation texts (DP–CT).
* LexRank is computationally intensive and so was not run on the DP-PA dataset (about 4000 sentences).

	Human Performance: ROUGE-2				
	human1	human2	human3	human4	average
Input: QA citation surveys					
QA–CT refs.	0.1807	0.1956	0.0756	0.2019	0.1635
QA–AB refs.	0.1116	0.1399	0.0711	0.1576	0.1201
Input: QA abstract surveys					
QA–CT refs.	0.1315	0.1104	0.1216	0.1151	0.1197
QA-AB refs.	0.2648	0.1977	0.1802	0.2544	0.2243
Input: DP citation surveys					
DP–CT refs.	0.1550	0.1259	0.1200	0.1654	0.1416

Table 4: ROUGE-2 scores obtained for each of the manually created surveys by using the other three as reference. ROUGE-1 and ROUGE-L followed similar patterns.

tracting the overlapping survey-worthy information across the two sources.

6.1.2 ROUGE evaluation

Table 4 presents ROUGE scores (Lin, 2004) of each of human-generated 250-word surveys against each other. The average (last column) is what the automatic surveys can aim for. We then evaluated each of the random surveys and those generated by the four summarization systems against the references. Table 5 lists ROUGE scores of surveys when the manually created 250-word survey of the QA citation texts, survey of the QA abstracts, and the survey of the DP citation texts, were used as gold standard.

When we use manually created citation text surveys as reference, then the surveys generated from citation texts obtained significantly better ROUGE scores than the surveys generated from abstracts and full papers ($p < 0.05$) [RESULT 1]. This shows that crucial survey-worthy information present in citation texts is not available, or hard to extract, from abstracts and papers alone. Further, the surveys generated from abstracts performed significantly better than those generated from the full papers ($p < 0.05$) [RESULT 2]. This shows that abstracts and citation texts are generally denser in survey worthy information than full papers.

When we use manually created abstract surveys as reference, then the surveys generated from abstracts obtained significantly better ROUGE scores than the surveys generated from citation texts and full papers ($p < 0.05$) [RESULT 3]. Further, and more importantly, the surveys generated from citation texts performed significantly better than those generated from the full papers ($p < 0.05$) [RESULT 4]. Again, this shows that abstracts and citation texts are richer in survey-worthy information. These results also show that abstracts of papers and citation texts have some overlapping information (RESULT 2 and RESULT 4), but they also have a significant amount of unique survey-worthy information (RESULT 1 and RESULT 3).

Among the automatic summarizers, C-LexRank and LexRank perform best. This is unlike the results found through the nugget-evaluation method, where Trimmer performed best. This suggests that Trim-

	System Performance: ROUGE-2				
	Random	C-LexRank	C-RR	LexRank	Trimmer
Input: QA citation surveys					
QA–CT refs.	0.11561	**0.17013**	0.09522	0.13501	0.16984
QA–AB refs.	0.08264	**0.11653**	0.07600	0.07013	0.10336
Input: QA abstract surveys					
QA–CT refs.	0.04516	0.05892	**0.06149**	0.05369	0.04114
QA–AB refs.	0.12085	0.13634	0.12190	**0.20311**	0.13357
Input: QA full paper surveys					
QA–CT refs.	0.03042	0.03606	0.03599	**0.28244**	0.03986
QA–AB refs.	0.04621	0.05901	0.04976	**0.10540**	0.07505
Input: DP citation surveys					
DP–CT refs.	0.10690	**0.13164**	0.08748	0.04901	0.10052
Input: DP abstract surveys					
DP–CT refs.	0.07027	0.07321	0.05318	**0.20311**	0.07176
Input: DP full paper surveys					
DP–CT refs.	0.03770	0.02511	0.03433	*	**0.04554**

Table 5: ROUGE-2 scores of automatic surveys of QA and DP data. The surveys are evaluated by using human references created from QA citation texts (QA–CT), QA abstracts (QA–AB), and DP citation texts (DP–CT). These results are obtained after Jack-knifing the human references so that the values can be compared to those in Table 4.
* LexRank is computationally intensive and so was not run on the DP full papers set (about 4000 sentences).

mer is better at identifying more useful nuggets of information, but C-LexRank and LexRank are better at producing unigrams and bigrams expected in a survey. To some extent this may be due to the fact that Trimmer uses smaller (trimmed) fragments of source sentences in its summaries.

7 Conclusion

In this paper, we investigated the usefulness of directly summarizing citation texts (sentences that cite other papers) in the automatic creation of technical surveys. We generated surveys of a set of Question Answering (QA) and Dependency Parsing (DP) papers, their abstracts, and their citation texts using four state-of-the-art summarization systems (C-LexRank, C-RR, LexRank, and Trimmer). We then used two separate approaches, nugget-based pyramid and ROUGE, to evaluate the surveys. The results from both approaches and all four summarization systems show that both citation texts and abstracts have unique survey-worthy information. These results also demonstrate that, unlike single document summarization (where citing sentences have been suggested to be inappropriate (Teufel et al., 2006)), multidocument summarization—especially technical survey creation—benefits considerably from citation texts.

We next plan to generate surveys using both citation texts and abstracts together as input. Given the overlapping content of abstracts and citation texts, discovered in the current study, it is clear that redundancy detection will be an integral component of this future work. Creating readily consumable surveys is a hard task, especially when using only raw text and simple summarization techniques. Therefore we intend to combine these summarization and bibliometric techniques with suitable visualization methods towards the creation of iterative technical survey tools—systems that present surveys and bibliometric links in a visually convenient manner and which incorporate user feedback to produce even better surveys.

Acknowledgments

This work was supported, in part, by the National Science Foundation under Grant No. IIS-0705832 (iOPENER: Information Organization for PENning Expositions on Research) and Grant No. 0534323 (Collaborative Research: BlogoCenter - Infrastructure for Collecting, Mining and Accessing Blogs), in part, by the Human Language Technology Center of Excellence, and in part, by the Center for Advanced Study of Language (CASL). 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 sponsors.

References

Shannon Bradshaw. 2003. Reference directed indexing: Redeeming relevance for subject search in citation indexes. In *Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries*.

Jaime G. Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336, Melbourne, Australia.

Aaron Elkiss, Siwei Shen, Anthony Fader, Güneş Erkan, David States, and Dragomir R. Radev. 2008a. Blind men and elephants: What do citation summaries tell us about a research article? *Journal of the American Society for Information Science and Technology*, 59(1):51–62.

Aaron Elkiss, Siwei Shen, Anthony Fader, Güneş Erkan, David States, and Dragomir R. Radev. 2008b. Blind men and elephants: What do citation summaries tell us about a research article? *Journal of the American Society for Information Science and Technology*, 59(1):51–62.

Güneş Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research*.

Wesley Hildebrandt, Boris Katz, and Jimmy Lin. 2004. Overview of the trec 2003 question-answering track. 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)*.

Mark Joseph and Dragomir Radev. 2007. Citation analysis, centrality, and the ACL Anthology. Technical Report CSE-TR-535-07, University of Michigan. Dept. of Electrical Engineering and Computer Science.

Daniel Jurafsky and James H. Martin. 2008. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics (2nd edition)*. Prentice-Hall.

Min-Yen Kan, Judith L. Klavans, and Kathleen R. McKeown. 2002. Using the Annotated Bibliography as a Resource for Indicative Summarization. In *Proceedings of LREC 2002*, Las Palmas, Spain.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of ACL*, pages 423–430.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *SIGIR '95*, pages 68–73, New York, NY, USA. ACM.

Amy Langville and Carl Meyer. 2006. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press.

Jimmy J. Lin and Dina Demner-Fushman. 2006. Methods for automatically evaluating answers to complex questions. *Information Retrieval*, 9(5):565–587.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the ACL workshop on Text Summarization Branches Out*.

Qiaozhu Mei and ChengXiang Zhai. 2008. Generating impact-based summaries for scientific literature. In *Proceedings of ACL '08*, pages 816–824.

Preslav I. Nakov, Schwartz S. Ariel, and Hearst A. Marti. 2004. Citances: Citation sentences for semantic analysis of bioscience text. In *Workshop on Search and Discovery in Bioinformatics*.

Hidetsugu Nanba and Manabu Okumura. 1999. Towards multi-paper summarization using reference information. In *IJCAI1999*, pages 926–931.

Hidetsugu Nanba, Takeshi Abekawa, Manabu Okumura, and Suguru Saito. 2004a. Bilingual presri: Integration of multiple research paper databases. In *Proceedings of RIAO 2004*, pages 195–211, Avignon, France.

Hidetsugu Nanba, Noriko Kando, and Manabu Okumura. 2004b. Classification of research papers using citation links and citation types: Towards automatic review article generation. In *Proceedings of the 11th SIG Classification Research Workshop*, pages 117–134, Chicago, USA.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. *Proceedings of the HLT-NAACL conference*.

Mark E. J. Newman. 2001. The structure of scientific collaboration networks. *PNAS*, 98(2):404–409.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *COLING 2008*, Manchester, UK.

Advaith Siddharthan and Simone Teufel. 2007. Whose idea was this, and why does it matter? attributing scientific work to citations. In *Proceedings of NAACL/HLT-07*.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Comput. Linguist.*, 28(4):409–445.

Simone Teufel, Advaith Siddharthan, and Dan Tidhar. 2006. Automatic classification of citation function. In *Proceedings of EMNLP*, pages 103–110, Australia.

Simone Teufel. 2005. Argumentative Zoning for Improved Citation Indexing. *Computing Attitude and Affect in Text: Theory and Applications*, pages 159–170.

Ellen M. Voorhees. 2003. Overview of the trec 2003 question answering track. In *Proceedings of the Twelfth Text Retrieval Conference (TREC 2003)*.

David M. Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2007. Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. *Information Processing and Management (Special Issue on Summarization)*.

Non-Parametric Bayesian Areal Linguistics

Hal Daumé III
School of Computing
University of Utah
Salt Lake City, UT 84112
me@hal3.name

Abstract

We describe a statistical model over linguistic areas and phylogeny. Our model recovers known areas and identifies a plausible hierarchy of areal features. The use of areas improves genetic reconstruction of languages both qualitatively and quantitatively according to a variety of metrics. We model linguistic areas by a Pitman-Yor process and linguistic phylogeny by Kingman's coalescent.

1 Introduction

Why are some languages more alike than others? This question is one of the most central issues in historical linguistics. Typically, one of three answers is given (Aikhenvald and Dixon, 2001; Campbell, 2006). First, the languages may be related "genetically." That is, they may have all derived from a common ancestor language. Second, the similarities may be due to chance. Some language properties are simply more common than others, which is often attributed to be mostly due to linguistic universals (Greenberg, 1963). Third, the languages may be related *areally*. Languages that occupy the same geographic area often exhibit similar characteristics, not due to genetic relatedness, but due to sharing. Regions (and the languages contained within them) that exhibit sharing are called *linguistic areas* and the features that are shared are called *areal features*.

Much is not understood or agreed upon in the field of areal linguistics. Different linguists favor different defintions of what it means to be a linguistic area (are two languages sufficient to describe an area or do you need three (Thomason, 2001; Katz, 1975)?), what areal features are (is there a linear ordering of "borrowability" (Katz, 1975; Curnow, 2001) or is that too prescriptive?), and what causes sharing to take place (does social status or number of speakers play a role (Thomason, 2001)?).

In this paper, we attempt to provide a *statistical* answer to some of these questions. In particular, we develop a Bayesian model of typology that allows for, but does not force, the existence of linguistic areas. Our model also allows for, but does not force, preference for some feature to be shared areally. When applied to a large typological database of linguistic features (Haspelmath et al., 2005), we find that it discovers linguistic areas that are well documented in the literature (see Campbell (2005) for an overview), and a small preference for certain features to be shared areally. This latter agrees, to a lesser degree, with some of the published hierarchies of borrowability (Curnow, 2001). Finally, we show that reconstructing language family trees is significantly aided by knowledge of areal features. We note that Warnow et al. (2005) have independently proposed a model for phonological change in Indo-European (based on the Dyen dataset (Dyen et al., 1992)) that includes notions of borrowing. Our model is different in that we (a) base our model on typological features rather than just lexical patterns and (b) we explicitly represent language areas, not just one-time borrowing phenomena.

2 Background

We describe (in Section 3) a non-parametric, hierarchical Bayesian model for finding linguistic areas and areal features. In this section, we provide necessary background—both linguistic and statistical—

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 593–601,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

for understanding our model.

2.1 Areal Linguistics

Areal effects on linguistic typology have been studied since, at least, the late 1920s by Trubetzkoy, though the idea of tracing family trees for languages goes back to the mid 1800s and the comparative study of historical linguistics dates back, perhaps to Giraldus Cambrenis in 1194 (Campbell, In press). A recent article provides a short introduction to both the issues that surround areal linguistics, as well as an enumeration of many of the known language areas (Campbell, 2005). A fairly wide, modern treatment of the issues surrounding areal diffusion is also given by essays in a recent book edited by Aikhenvald and Dixon (2001). The essays in this book provide a good introduction to the issues in the field. Campbell (2006) provides a critical survey of these and other hypotheses relating to areal linguistics.

There are several issues which are basic to the study of areal linguistics (these are copied almost directly from Campbell (2006)). Must a linguistic area comprise more than two languages? Must it comprise more than one language family? Is a single trait sufficient to define an area? How "nearby" must languages in an area be to one another? Are some feature more easily borrowed that others?

Despite these formal definitional issues of what constitutes a language area and areal features, most historical linguists seem to believe that areal effects play *some* role in the change of languages.

2.1.1 Established Linguistic Areas

Below, we list some of the well-known linguistic areas; Campbell (2005) provides are more complete listing together with example areal features for these areas. For each area, we list associated languages:

The Balkans: Albanian, Bulgarian, Greek, Macedonian, Rumanian and Serbo-Croatian. (*Sometimes:* Romani and Turkish)

South Asian: Languages belonging to the Dravidian, Indo-Aryan, Munda, Tibeto-Burman families.

Meso-America: Cuitlatec, Huave, Mayan, Mixe-Zoquean, Nahua, Otomanguean, Tarascan, Tequistlatecan, Totonacan and Xincan.

North-west America: Alsea, Chimakuan, Coosan, Eyak, Haida, Kalapuyan, Lower Chinook, Salishan, Takelman, Tlingit, Tsimshian and Wakashan.

The Baltic: Baltic languages, Baltic German, and Finnic languages (especially Estonian and Livonian). (Sometimes many more are included, such as: Belorussian, Lavian, Lithuanian, Norwegian, Old Prussian, Polish, Romani, Russian, Ukranian.)

Ethiopia: Afar, Amharic, Anyuak, Awngi, Beja, Ge'ez, Gumuz, Janjero, Kefa, Sidamo, Somali, Tigre, Tigrinya and Wellamo.

Needless to say, the exact definition and extent of the actual areas is up to significant debate. Moreover, claims have been made in favor of many linguistic areas not defined above. For instance, Dixon (2001) presents arguments for several Australian linguistic areas and Matisoff (2001) defines a South-East Asian language area. Finally, although "folk lore" is in favor of identifying a linguistic area including English, French and certain Norse languages (Norwegian, Swedish, Low Dutch, High German, etc.), there are counter-arguments to this position (Thomason, 2001) (see especially Case Study 9.8).

2.1.2 Linguistic Features

Identifying which linguistic features are most easily shared "areally" is a long standing problem in contact linguistics. Here we briefly review some of the major claims. Much of this overview is adoped from the summary given by Curnow (2001).

Haugen (1950) considers only borrowability as far as the lexicon is concerned. He provided evidence that nouns are the easiest, followed by verbs, adjectives, adverbs, prepositions, etc. Ross (1988) corroborates Haugen's analysis and deepens it to cover morphology, syntax and phonology. He proposes the following hierarchy of borrowability (easiest items coming first): nouns $>$ verbs $>$ adjectives $>$ syntax $>$ non-bound function words $>$ bound morphemes $>$ phonemes. Coming from a "constraints" perspective, Moravcsik (1978) suggests that: lexical items must be borrowed before lexical properties; inflected words before bound morphemes; verbal items can never be borrowed; etc.

Curnow (2001) argues that coming up with a reasonable hierarchy of borrowability is that "we may never be able to develop such constraints." Nevertheless, he divides the space of borrowable features into 15 categories and discusses the evidence supporting each of these categories, including: phonetics (rare), phonology (common), lexical (very common), interjections and discourse markers (com-

mon), free grammatical forms (occasional), bound grammatical forms (rare), position of morphology (rare), syntactic frames (rare), clause-internal syntax (common), between-clause syntax (occasional).

2.2 Non-parametric Bayesian Models

We treat the problem of understanding areal linguistics as a statistical question, based on a database of typological information. Due to the issues raised in the previous section, we do not want to commit to the existence of a particular number of linguistic areas, or particular sizes thereof. (Indeed, we do not even want to commit to the existence of *any* linguistic areas.) However, we will need to "unify" the languages that fall into a linguistic area (if such a thing exists) by means of some statistical parameter. Such problems have been studied under the name *non-parametric models*. The idea behind non-parametric models is that one does not commit *a priori* to a particularly number of parameters. Instead, we allow the data to dictate how many parameters there are. In Bayesian modeling, non-parametric distributions are typically used as *priors*; see Jordan (2005) or Ghahramani (2005) for overviews. In our model, we use two different non-parametric priors: the Pitman-Yor process (for modeling linguistic areas) and Kingman's coalescent (for modeling linguistic phylogeny), both described below.

2.2.1 The Pitman-Yor Process

One particular example of a non-parametric prior is the Pitman-Yor process (Pitman and Yor, 1997), which can be seen as an extension to the better-known Dirichlet process (Ferguson, 1974). The Pitman-Yor process can be understood as a particular example of a Chinese Restaurant process (CRP) (Pitman, 2002). The idea in all CRPs is that there exists a restaurant with an infinite number of tables. Customers come into the restaurant and have to choose a table at which to sit.

The Pitman-Yor process is described by three parameters: a base rate α, a discount parameter d and a mean distribution G_0. These combine to describe a process denoted by $\mathcal{PY}(\alpha, d, G_0)$. The parameters α and d must satisfy: $0 \leq d < 1$ and $\alpha > -d$. In the CRP analogy, the model works as follows. The first customer comes in and sits at any table. After N customers have come in and seated themselves (at a total of K tables), the Nth customer arrives. In the Pitman-Yor process, the Nth customer sits at a new table with probability proportional to $\alpha + Kd$ and sits at a previously occupied table k with probability proportional to $\#_k - d$, where $\#_k$ is the number of customers already seated at table k. Finally, with each table k we associate a parameter θ_k, with each θ_k drawn independently from G_0. An important property of the Pitman-Yor process is that draws from it are *exchangable*: perhaps counterintuitively, the distribution does not care about customer order.

The Pitman-Yor process induces a power-law distribution on the number of singleton tables (i.e., the number of tables that have only one customer). This can be seen by noticing two things. In general, the number of singleton tables grows as $\mathcal{O}(\alpha N^d)$. When $d = 0$, we obtain a Dirichlet process with the number of singleton tables growing as $\mathcal{O}(\alpha \log N)$.

2.2.2 Kingman's Coalescent

Kingman's coalescent is a standard model in population genetics describing the common genealogy (ancestral tree) of a set of individuals (Kingman, 1982b; Kingman, 1982a). In its full form it is a distribution over the genealogy of a countable set.

Consider the genealogy of n individuals alive at the present time $t = 0$. We can trace their ancestry backwards in time to the distant past $t = -\infty$. Assume each individual has one parent (in genetics, *haploid* organisms), and therefore genealogies of $[n] = \{1, \ldots, n\}$ form a *directed forest*. Kingman's n-coalescent is simply a distribution over genealogies of n individuals. To describe the Markov process in its entirety, it is sufficient to describe the jump process (i.e. the embedded, discrete-time, Markov chain over partitions) and the distribution over coalescent times. In the n-coalescent, every pair of lineages merges independently with rate 1, with parents chosen uniformly at random from the set of possible parents at the previous time step.

The n-coalescent has some interesting statistical properties (Kingman, 1982b; Kingman, 1982a). The marginal distribution over tree topologies is uniform and independent of the coalescent times. Secondly, it is infinitely exchangeable: given a genealogy drawn from an n-coalescent, the genealogy of any m contemporary individuals alive at time $t \leq 0$ embedded within the genealogy is a draw from the m-coalescent. Thus, taking $n \to \infty$, there is a distri-

bution over genealogies of a countably infinite population for which the marginal distribution of the genealogy of any n individuals gives the n-coalescent. Kingman called this *the coalescent*.

Teh et al. (2007) recently described efficient inference algorithms for Kingman's coalescent. They applied the coalescent to the problem of recovering linguistic phylogenies. The application was largely successful—at least in comparison to alternative algorithms that use the same data-. Unfortunately, even in the results they present, one can see significant areal effects. For instance, in their Figure(3a), Romanian is very near Albanian and Bulgarian. This is likely an areal effect: specifically, an effect due to the Balkan langauge area. We will revisit this issue in our own experiments.

3 A Bayesian Model for Areal Linguistics

We will consider a data set consisting of N languages and F typological features. We denote the value of feature f in language n as $X_{n,f}$. For simplicity of exposition, we will assume two things: (1) there is no unobserved data and (2) all features are binary. In practice, for the data we use (described in Section 4), neither of these is true. However, both extensions are straightforward.

When we construct our model, we attempt to be as neutral to the "areal linguistics" questions defined in Section 2.1 as possible. We allow areas with only two languages (though for brevity we do not present them in the results). We allow areas with only one family (though, again, do not present them). We are generous with our notion of locality, allowing a radius of 1000 kilometers (though see Section 5.4 for an analysis of the effect of radius).[1] And we allow, but do not enforce trait weights. All of this is accomplished through the construction of the model and the choice of the model hyperparameters.

At a high-level, our model works as follows. Values $X_{n,f}$ appear for one of two reasons: they are either areally derived or genetically derived. A latent variable $Z_{n,f}$ determines this. If it is derived areally, then the value $X_{n,f}$ is drawn from a latent variable corresponding to the value preferences in the langauge area to which language n belongs. If it is derived genetically, then $X_{n,f}$ is drawn from a variable corresponding to value preferences for the genetic substrate to which language n belongs. The set of areas, and the area to which a language belongs are given by yet more latent variables. It is this aspect of the model for which we use the Pitman-Yor process: languages are customers, areas are tables and area value preferences are the parameters of the tables.

3.1 The formal model

We assume that the value a feature takes for a particular language (i.e., the value of $X_{n,f}$) can be explained *either* genetically or areally.[2] We denote this by a binary indicator variable $Z_{n,f}$, where a value 1 means "areal" and a value 0 means "genetic." We assume that each $Z_{n,f}$ is drawn from a feature-specific binomial parameter π_f. By having the parameter feature-specific, we express the fact that some features may be more or less likely to be shared than others. In other words, a high value of π_f would mean that feature f is easily shared areally, while a low value would mean that feature f is hard to share. Each language n has a known latitude/longitude ℓ_n.

We further assume that there are K linguistic areas, where K is treated non-parametrically by means of the Pitman-Yor process. Note that in our context, a linguistic area may contain *only one* language, which would technically not be allowed according to the linguistic definition. When a language belongs to a singleton area, we interpret this to mean that it does not belong to any language area.

Each language area k (including the singleton areas) has a set of F associated parameters $\phi_{k,f}$, where $\phi_{k,f}$ is the probability that feature f is "on" in area k. It also has a "central location" given by a longitude and latitude denoted c_k. We only allow languages to belong to areas that fall within a given radius R of them (distances computed according to geodesic distance). This accounts for the "geographical" constraints on language areas. We denote the area to which language n belongs as a_n.

We assume that each language belongs to a "family tree." We denote the parent of language n in the

[1] An reader might worry about exchangeability: Our method of making language centers and locations part of the Pitman-Yor distribution ensures this is not an issue. An alternative would be to use a location-sensitive process such as the kernel stick-breaking process (Dunson and Park, 2007), though we do not explore that here.

[2] As mentioned in the introduction, (at least) one more option is possible: chance. We treat "chance" as noise and model it in the data generation process, not as an alternative "source."

$X_{n,f} \sim \begin{cases} \mathcal{B}in(\theta_{p_n,f}) & \text{if } Z_{n,f}=0 \\ \mathcal{B}in(\phi_{a_n,f}) & \text{if } Z_{n,f}=1 \end{cases}$	feature values are derived genetically or areally
$Z_{n,f} \sim \mathcal{B}in(\pi_f)$	feature source is a biased coin, parameterized per feature
$\ell_n \sim \mathcal{B}all(c_{a_n}, R)$	language position is uniform within a ball around area center, radius R
$\pi_f \sim \mathcal{B}et(1,1)$	bias for a feature being genetic/areal is uniform
$(p,\theta) \sim \text{Coalescent}(\pi_0, m_0)$	language hierarchy and genetic traits are drawn from a Coalescent
$(a,\langle\phi,c\rangle) \sim \mathcal{PY}(\alpha_0, d_0, \mathcal{B}et(1,1)\times\mathcal{U}ni)$	area features are drawn Beta and centers Uniformly across the globe

Figure 1: Full hierarchical Areal model; see Section 3.1 for a complete description.

family tree by p_n. We associate with each node i in the family tree and each feature f a parameter $\theta_{i,f}$. As in the areal case, $\theta_{i,f}$ is the probability that feature f is on for languages that descend from node i in the family tree. We model genetic trees by Kingman's coalescent with binomial mutation.

Finally, we put non-informative priors on all the hyperparameters. Written hierarchically, our model has the following shown in Figure 1. There, by $(p,\theta) \sim \text{Coalescent}(\pi_0, m_0)$, we mean that the tree and parameters are given by a coalescent.

3.2 Inference

Inference in our model is mostly by Gibbs sampling. Most of the distributions used are conjugate, so Gibbs sampling can be implemented efficiently. The only exceptions are: (1) the coalescent for which we use the GreedyRate1 algorithm described by Teh et al. (2007); (2) the area centers c, for which we using a Metropolis-Hastings step. Our proposal distribution is a Gaussian centered at the previous center, with standard deviation of 5. Experimentally, this resulted in an acceptance rate of about 50%.

In our implementation, we analytically integrate out π and ϕ and sample only over Z, the coalescent tree, and the area assignments. In some of our experiments, we treat the family tree as given. In this case, we also analytically integrate out the θ parameters and sample only over Z and area assignments.

4 Typological Data

The database on which we perform our analysis is the *World Atlas of Language Structures* (henceforth, WALS) (Haspelmath et al., 2005). The database contains information about 2150 languages (sampled from across the world). There are 139 typological *features* in this database. The database is *sparse*: only 16% of the possible language/feature pairs are known. We use the version extracted and preprocessed by Daumé III and Campbell (2007).

In WALS, languages a grouped into 38 language families (including Indo-European, Afro-Asiatic, Austronesian, Niger-Congo, etc.). Each of these language families is grouped into a number of language geni. The Indo-European family includes ten geni, including: Germanic, Romance, Indic and Slavic. The Austronesian family includes seventeen geni, including: Borneo, Oceanic, Palauan and Sundic. Overall, there are 275 geni represented in WALS.

We further preprocess the data as follows. For the Indo-European subset (hence-forth, "IE"), we remove all languages with ≤ 10 known features and then remove all features that appear in at most $1/4$ of the languages. This leads to 73 languages and 87 features. For the whole-world subset, we remove languages with ≤ 25 known features and then features that appear in at most $1/10$ of the languages. This leads to 349 languages and 129 features.

5 Experiments

5.1 Identifying Language Areas

Our first experiment is aimed at discovering language areas. We first focus on the IE family, and then extend the analysis to all languages. In both cases, we use a known family tree (for the IE experiment, we use a tree given by the language genus structure; for the whole-world experiment, we use a tree given by the language family structure). We run each experiment with five random restarts and 2000 iterations. We select the MAP configuration from the combination of these runs.

In the IE experiment, the model identified the areas shown in Figure 5.1. The best area identified by our model is the second one listed, which clearly correlates highly with the Balkans. There are two areas identified by our model (the first and last) that include only Indic and Iranian languages. While we are not aware of previous studies of these as linguistic areas, they are not implausible given

(Indic) Bhojpuri, Darai, Gujarati, Hindi, Kalami, Kashmiri, Kumauni, Nepali, Panjabi, Shekhawati, Sindhi **(Iranian)** Ormuri, Pashto
(Albanian) Albanian **(Greek)** Greek (Modern) **(Indic)** Romani (Kalderash) **(Romance)** Romanian, Romansch (Scharans), Romansch (Sursilvan), Sardinian **(Slavic)** Bulgarian, Macedonian, Serbian-Croatian, Slovak, Slovene, Sorbian
(Baltic) Latvian, Lithuanian **(Germanic)** Danish, Swedish **(Slavic)** Polish, Russian
(Celtic) Irish **(Germanic)** English, German, Norwegian **(Romance)** French
(Indic) Prasuni, Urdu **(Iranian)** Persian, Tajik
Plus 46 non-areal languages

Figure 2: IE areas identified. Areas that consist of just one genus are not listed, nor are areas with two languages.

(Mayan) Huastec, Jakaltek, Mam, Tzutujil **(Mixe-Zoque)** Zoque (Copainalá) **(Oto-Manguean)** Mixtec (Chalcatongo), Otomí (Mezquital) **(Uto-Aztecan)** Nahualtl (Tetelcingo), Pipil
(Baltic) Latvian, Lithuanian **(Finnic)** Estonian, Finnish **(Slavic)** Polish, Russian, Ukranian
(Austro-Asiatic) Khasi **(Dravidian)** Telugu **(IE)** Bengali **(Sino-Tibetan)** Bawm, Garo, Newari (Kathmandu)

Figure 3: A small subset of the world areas identified.

the history of the region. The fourth area identified by our model corresponds roughly to the debated "English" area. Our area includes the requisite French/English/German/Norwegian group, as well as the somewhat surprising Irish. However, in addition to being intuitively plausible, it is not hard to find evidence in the literature for the contact relationship between English and Irish (Sommerfelt, 1960).

In the whole-world experiment, the model identified too many linguistic areas to fit (39 in total that contained at least two languages, and contained at least two language families). In Figure 5.1, we depict the areas found by our model that best correspond to the areas described in Section 2.1.1. We acknowledge that this gives a warped sense of the quality of our model. Nevertheless, our model *is* able to identify large parts of the the Meso-American area, the Baltic area and the South Asian area. (It also finds the Balkans, but since these languages are all IE, we do not consider it a linguistic area in this evaluation.) While our model does find areas that match Meso-American and North-west American areas, neither is represented in its entirety (according to the definition of these areas given in Section 2.1.1).

Model	Rand	F-Sc	Edit	NVI
K-means	0.9149	0.0735	0.1856	0.5889
Pitman-Yor	0.9637	0.1871	0.6364	0.7998
Areal model	**0.9825**	**0.2637**	**0.8295**	**0.9090**

Table 1: Area identification scores for two baseline algorithms (K-means and Pitman-Yor clustering) that do not use hierarchical structure, and for the Areal model we have presented. Higher is better and all differences are statistically significant at the 95% level.

Despite the difficulty humans have in assigning linguistic areas, In Table 1, we explicitly compare the quality of the areal clusters found on the IE subset. We compare against the most inclusive areal lists from Section 2.1.1 for IE: the Balkans and the Baltic. When there is overlap (eg., Romani appears in both lists), we assigned it to the Balkans.

We compare our model with a flat Pitman-Yor model that does not use the hierarchy. We also compare to a baseline K-means algorithm. For K-means, we ran with $K \in \{5, 10, 15, \ldots, 80, 85\}$ and chose the value of K for each metric that did best (giving an unfair advantage). Clustering performance is measured on the Indo-European task according to the Rand Index, F-score, Normalized Edit Score (Pantel, 2003) and Normalized Variation of Information (Meila, 2003). In these results, we see that the Pitman-Yor process model dominates the K-means model and the Areal model dominates the Pitman-Yor model.

5.2 Identifying Areal Features

Our second experiment is an analysis of the *features* that tend to be shared areally (as opposed to genetically). For this experiment, we make use of the whole-world version of the data, again with known language family structure. We initialize a Gibbs sampler from the MAP configuration found in Section 5.1. We run the sampler for 1000 iterations and take samples every ten steps.

From one particular sample, we can estimate a posterior distribution over each π_f. Due to conjugacy, we obtain a posterior distribution of $\pi_f \sim \mathcal{B}et(1 + \sum_n Z_{n,f}, 1 + \sum_n [1 - Z_{n,f}])$. The 1s come from the prior. From this Beta distribution, we can ask the question: what is the probability that a value of π_f drawn from this distribution will have value < 0.5? If this value is high, then the feature is likely

p(gen)	#f	Feature Category
.00	1	Tea
.73	19	Phonology
.73	9	Lexicon
.74	4	Nominal Categories / Numerals
.79	5	Simple Clauses / Predication
.80	5	Verbal Categories / Tense and Aspect
.87	8	Nominal Syntax
.87	8	Simple Clauses / Simple Clauses
.91	12	Nominal Categories / Articles and Pronouns
.94	17	Word Order
.99	10	Morphology
.99	6	Simple Clauses / Valence and Voice
.99	7	Complex Sentences
.99	7	Nominal Categories / Gender and Number
.99	5	Simple Clauses / Negation and Questions
1.0	1	Other / Clicks
1.0	2	Verbal Categories / Suppletion
1.0	9	Verbal Categories / Modality
1.0	4	Nominal Categories / Case

Table 2: Average probability of genetic for each feature category and the number of features in that category.

to be a "genetic feature"; if it is low, then the feature is likely to be an "areal feature." We average these probabilities across all 100 samples.

The features that are *most* likely to be areal according to our model are summaries in Table 2. In this table, we list the *categories* to which each feature belongs, together with the number of features in that category, and the *average* probability that a feature in that category is genetically transmitted. Apparently, the vast majority of features are *not* areal.

We can treat the results presented in Table 2 as a hierarchy of borrowability. In doing so, we see that our hierarchy agrees to a large degree with the hierarchies summarized in Section 2.1.2. Indeed, (aside from "Tea", which we will ignore) the two most easily shared categories according to our model are phonology and the lexicon; this is in total agreement with the agreed state of affairs in linguistics.

Lower in our list, we see that noun-related categories tend to precede their verb-related counterparts (nominal categories before verbal categores, nominal syntax before complex sentences). According to Curnow (2001), the most difficult features to borrow are phonetics (for which we have no data), bound grammatical forms (which appear low on our list), morphology (which is 99% genetic, according to our model) and syntactic frames (which would roughly correspond to "complex sentences", another item which is 99% genetic in our model).

Indo-European

Model	Accuracy	Log Prob
Baseline	0.635 (± 0.007)	-0.583 (± 0.008)
Areal model	$\mathbf{0.689}$ (± 0.010)	$\mathbf{-0.526}$ (± 0.027)

World

Model	Accuracy	Log Prob
Baseline	0.628 (± 0.001)	-0.654 (± 0.003)
Areal model	$\mathbf{0.635}$ (± 0.002)	$\mathbf{-0.565}$ (± 0.011)

Table 3: Prediction accuracies and log probabilities for IE (top) and the world (bottom).

5.3 Genetic Reconstruction

In this section, we investigate whether the use of areal knowledge can improve the automatic reconstruction of language family trees. We use Kingman's coalescent (see Section 2.2.2) as a probabilistic model of trees, endowed with a binomial mutation process on the language features.

Our baseline model is to run the vanilla coalescent on the WALS data, effective reproducing the results presented by Teh et al. (2007). This method was already shown to outperform competing hierarchical clustering algorithms such as average-link agglomerative clustering (see, eg., Duda and Hart (1973)) and the Bayesian Hierarchical Clustering algorithm (Heller and Ghahramani, 2005).

We run the same experiment both on the IE subset of data and on the whole-world subset. We evaluate the results qualitatively, by observing the trees found (on the IE subset) and quantitatively (below). For the qualitative analysis, we show the subset of IE that does not contain Indic languages or Iranian languages (just to keep the figures small). The tree derived from the original data is on the left in Figure 4, below:
The tree based on areal information is on the right in Figure 4, below. As we can see, the use of areal information qualitatively improves the structure of the tree. Where the original tree had a number of errors with respect to Romance and Germanic languages, these are sorted out in the areally-aware tree. Moreover, Greek now appears in a more appropriate part of the tree and English appears on a branch that is further out from the Norse languages.

We perform two varieties of quantitative analysis. In the first, we attempt to predict unknown feature values. In particular, we *hide* an addition 10% of the feature values in the WALS data and fit a model

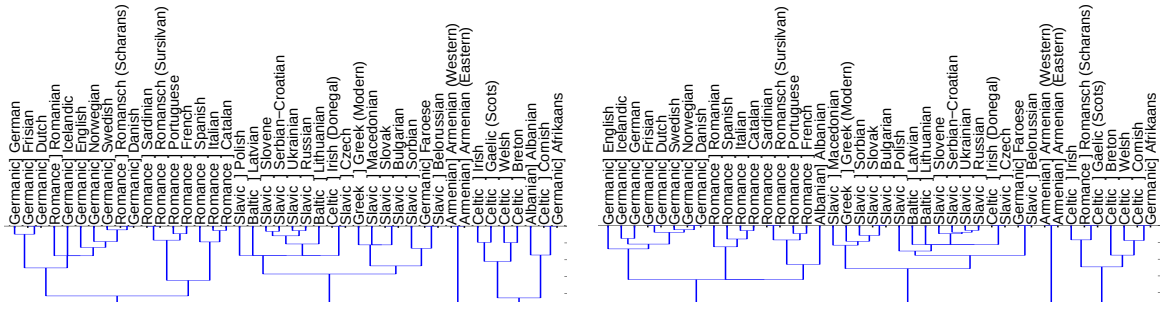


Figure 4: Genetic trees of IE languages. (Left) with no areal knowledge; (Right) with areal model.

Indo-European versus Genus

Model	Purity	Subtree	LOO Acc
Baseline	0.6078	0.5065	**0.3218**
Areal model	**0.6494**	**0.5455**	0.2528

World versus Genus

Model	Purity	Subtree	LOO Acc
Baseline	0.3599	0.2253	0.7747
Areal model	**0.4001**	**0.2450**	**0.7982**

World versus Family

Model	Purity	Subtree	LOO Acc
Baseline	0.4163	0.3280	0.4842
Areal model	**0.5143**	**0.3318**	**0.5198**

Table 4: Scores for IE as compared against genus (top); for world against genus (mid) and against family (low).

to the remaining 90%. We then use that model to predict the hidden 10%. The baseline model is to make predictions according to the family tree. The augmented model is to make predictions according to the family tree *for those features identified as genetic* and according to the linguistic area *for those features identified as areal.* For both settings, we compute both the absolute accuracy as well as the log probability of the hidden data under the model (the latter is less noisy). We repeat this experiment 10 times with a different random 10% hidden. The results are shown in Table 3, below. The differences are not large, but are outside one standard deviation.

For the second quantitative analysis, we use present purity scores (Heller and Ghahramani, 2005), subtree scores (the number of interior nodes with pure leaf labels, normalized) and leave-one-out log accuracies (all scores are between 0 and 1, and higher scores are better). These scores are computed against both language family and language genus as the "classes." The results are in Table 4, below. As we can see, the results are generally in favor of the Areal model (LOO Acc on IE versus Genus notwithstanding), depending on the evaluation metric.

Radius	Purity	Subtree	LOO Acc
125	0.6237	0.4855	0.2013
250	0.6457	0.5325	0.2299
500	0.6483	**0.5455**	0.2413
1000	**0.6494**	**0.5455**	0.2528
2000	0.6464	0.4935	0.3218
4000	0.6342	0.4156	**0.4138**

Table 5: Scores for IE vs genus at varying radii.

5.4 Effect of Radius

Finally, we evaluate the effect of the radius hyperparameter on performance. Table 5 shows performance for models built with varying radii. As can be seen by purity and subtree scores, there is a "sweet spot" around 500 to 1000 kilometers where the model seems optimal. LOO (strangely) seems to continue to improve as we allow areas to grow arbitrarily large. This is perhaps overfitting. Nevertheless, performance is robust for a range of radii.

6 Discussion

We presented a model that is able to recover well-known linguistic areas. Using this areas, we have shown improvement in the ability to recover phylogenetic trees of languages. It is important to note that despite our successes, there is much at our model does not account for: borrowing is known to be assymetric; contact is temporal; borrowing must obey univeral implications. Despite the failure of our model to account for these issues, however, it appears largely successful. Moreover, like any "data mining" expedition, our model suggests new linguistic areas (particularly in the "whole world" experiments) that deserve consideration.

Acknowledgments

Deep thanks to Lyle Campbell, Yee Whye Teh and Eric Xing for discussions; comments from the three anonymous reviewers were very helpful. This work was partially supported by NSF grant IIS0712764.

References

Alexandra Aikhenvald and R.M.W. Dixon, editors. 2001. *Areal diffusion and genetic inheritance: problems in comparative linguistics*. Oxford University Press.

Lyle Campbell. 2005. Areal linguistics. In Keith Brown, editor, *Encyclopedia of Language and Linguistics*. Elsevier, 2 edition.

Lyle Campbell. 2006. Areal linguistics: the problem to the answer. In April McMahon, Nigel Vincent, and Yaron Matras, editors, *Language contact and areal linguistics*.

Lyle Campbell. In press. Why Sir William Jones got it all wrong, or Jones' role in how to establish language families. In Joseba Lakarra, editor, *Festschrift/Memorial volume for Larry Trask*.

Timothy Curnow. 2001. What language features can be "borrowed"? In Aikhenvald and Dixon, editors, *Areal diffusion and genetic inheritance: problems in comparative linguistics*, pages 412–436. Oxford University Press.

Hal Daumé III and Lyle Campbell. 2007. A Bayesian model for discovering typological implications. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

R.M.W. Dixon. 2001. The Australian linguistic area. In Aikhenvald and Dixon, editors, *Areal diffusion and genetic inheritance: problems in comparative linguistics*, pages 64–104. Oxford University Press.

R. O. Duda and P. E. Hart. 1973. *Pattern Classification And Scene Analysis*. Wiley and Sons, New York.

David Dunson and Ju-Hyun Park. 2007. Kernel stick breaking processes. *Biometrika*, 95:307–323.

Isidore Dyen, Joseph Kurskal, and Paul Black. 1992. An Indoeuropean classification: A lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5). American Philosophical Society.

Thomas S. Ferguson. 1974. Prior distributions on spaces of probability measures. *The Annals of Statistics*, 2(4):615–629, July.

Zoubin Ghahramani. 2005. Nonparametric Bayesian methods. Tutorial presented at UAI conference.

Joseph Greenberg, editor. 1963. *Universals of Languages*. MIT Press.

Martin Haspelmath, Matthew Dryer, David Gil, and Bernard Comrie, editors. 2005. *The World Atlas of Language Structures*. Oxford University Press.

E. Haugen. 1950. The analysis of linguistic borrowing. *Language*, 26:210–231.

Katherine Heller and Zoubin Ghahramani. 2005. Bayesian hierarchical clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 22.

Michael I. Jordan. 2005. Dirichlet processes, Chinese restaurant processes and all that. Tutorial presented at NIPS conference.

Harmut Katz. 1975. *Generative Phonologie und phonologische Sprachbünde des Ostjakischen un Samojedischen*. Wilhelm Fink.

J. F. C. Kingman. 1982a. The coalescent. *Stochastic Processes and their Applications*, 13:235–248.

J. F. C. Kingman. 1982b. On the genealogy of large populations. *Journal of Applied Probability*, 19:27–43. Essays in Statistical Science.

James Matisoff. 2001. Genetic versus contact relationship: prosodic diffusibility in South-East Asian languages. In Aikhenvald and Dixon, editors, *Areal diffusion and genetic inheritance: problems in comparative linguistics*, pages 291–327. Oxford University Press.

Marina Meila. 2003. Comparing clusterings. In *Proceedings of the Conference on Computational Learning Theory (COLT)*.

E. Moravcsik. 1978. Language contact. In J.H. Greenberg, C. Ferguson, and E. Moravcsik, editors, *Universals of Human Language*, volume 1; Method and Theory, pages 3–123. Stanford University Press.

Patrick Pantel. 2003. *Clustering by Committee*. Ph.D. thesis, University of Alberta.

J. Pitman and M. Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900.

Jim Pitman. 2002. Combinatorial stochastic processes. Technical Report 621, University of California at Berkeley. Lecture notes for St. Flour Summer School.

M.D. Ross. 1988. Proto Oceanic and the Austronesian languages of western melanesia. *Canberra: Pacific Linguitics, Australian National University*.

Alf Sommerfelt. 1960. External versus internal factors in the development of language. *Norsk Tidsskrift for Sprogvidenskap*, 19:296–315.

Yee Whye Teh, Hal Daumé III, and Daniel Roy. 2007. Bayesian agglomerative clustering with coalescents. In *Advances in Neural Information Processing Systems (NIPS)*.

Sarah Thomason. 2001. *Language contact: an introduction*. Edinburgh University Press.

T. Warnow, S.N. Evans, D. Ringe, and L. Nakhleh. 2005. A stochastic model of language evolution that incorporates homoplasy and borrowing. In *Phylogenetic Methods and the Prehistory of Language*. Cambridge University Press. Invited paper.

Hierarchical Bayesian Domain Adaptation

Jenny Rose Finkel and Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305
{jrfinkel|manning}@cs.stanford.edu

Abstract

Multi-task learning is the problem of maximizing the performance of a system across a number of related tasks. When applied to multiple domains for the same task, it is similar to domain adaptation, but symmetric, rather than limited to improving performance on a target domain. We present a more principled, better performing model for this problem, based on the use of a hierarchical Bayesian prior. Each domain has its own domain-specific parameter for each feature but, rather than a constant prior over these parameters, the model instead links them via a hierarchical Bayesian global prior. This prior encourages the features to have similar weights across domains, unless there is good evidence to the contrary. We show that the method of (Daumé III, 2007), which was presented as a simple "preprocessing step," is actually equivalent, except our representation explicitly separates hyperparameters which were tied in his work. We demonstrate that allowing different values for these hyperparameters significantly improves performance over both a strong baseline and (Daumé III, 2007) within both a conditional random field sequence model for named entity recognition and a discriminatively trained dependency parser.

1 Introduction

The goal of *multi-task learning* is to improve performance on a set of related tasks, when provided with (potentially varying quantities of) annotated data for each of the tasks. It is very closely related to *domain adaptation*, a far more common task in the natural language processing community, but with two primary differences. Firstly, in domain adaptation the different tasks are actually just different domains. Secondly, in multi-task learning the focus is on improving performance across *all* tasks, while in domain adaptation there is a distinction between *source* data and *target* data, and the goal is to improve performance on the target data. In the present work we focus on domain adaptation, but like the multi-task setting, we wish to improve performance across *all* domains and not a single *target* domains. The word *domain* is used here somewhat loosely: it may refer to a topical domain or to distinctions that linguists might term mode (speech versus writing) or register (formal written prose versus SMS communications). For example, one may have a large amount of parsed newswire, and want to use it to augment a much smaller amount of parsed e-mail, to build a higher quality parser for e-mail data. We also consider the extension to the task where the annotation is not the same, but is consistent, across domains (that is, some domains may be annotated with more information than others).

This problem is important because it is omnipresent in real life natural language processing tasks. Annotated data is expensive to produce and limited in quantity. Typically, one may begin with a considerable amount of annotated newswire data, some annotated speech data, and a little annotated e-mail data. It would be most desirable if the aggregated training data could be used to improve the performance of a system on each of these domains.

From the baseline of building separate systems for each domain, the obvious first attempt at domain adaptation is to build a system from the union of the training data, and we will refer to this as a second baseline. In this paper we propose a more principled, formal model of domain adaptation, which not only outperforms previous work, but maintains attractive

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 602–610,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

performance characteristics in terms of training and testing speed. We also show that the domain adaptation work of (Daumé III, 2007), which is presented as an ad-hoc "preprocessing step," is actually equivalent to our formal model. However, our representation of the model conceptually separates some of the hyperparameters which are not separated in (Daumé III, 2007), and we found that setting these hyperparameters with different values from one another was critical for improving performance.

We apply our model to two tasks, named entity recognition, using a linear chain conditional random field (CRF), and dependency parsing, using a discriminative, chart-based model. In both cases, we find that our model improves performance over both baselines and prior work.

2 Hierarchical Bayesian Domain Adaptation

2.1 Motivation

We call our model *hierarchical Bayesian domain adaptation*, because it makes use of a hierarchical Bayesian prior. As an example, take the case of building a logistic classifier to decide if a word is part of a person's name. There will be a parameter (weight) for each feature, and usually there is a zero-mean Gaussian prior over the parameter values so that they don't get too large.[1] In the standard, single-domain, case the log likelihood of the data and prior is calculated, and the optimal parameter values are found. Now, let's extend this model to the case of two domains, one containing American newswire and the other containing British newswire. The data distributions will be similar for the two domains, but not identical. In our model, we have separate parameters for each feature in each domain. We also have a top level parameter (also to be learned) for each feature. For each domain, the Gaussian prior over the parameter values is now centered around these top level parameters instead of around zero. A zero-mean Gaussian prior is then placed over the top level parameters. In this example, if some feature, say *word='Nigel,'* only appears in the British newswire, the corresponding weight for the American newswire will have a similar value. This happens because the evidence in the British domain will push the British parameter to have a high value, and this will in turn influence the top-level parameter to have a high value, which will then influence the American newswire to have a high value, because there will be no evidence in the American data to override the prior. Conversely, if some feature is highly indicative of *isName=true* for the British newswire, and of *isName=false* for the American newswire, then the British parameter will have a high (positive) value while the American parameter will have a low (negative) value, because in both cases the domain-specific evidence will outweigh the effect of the prior.

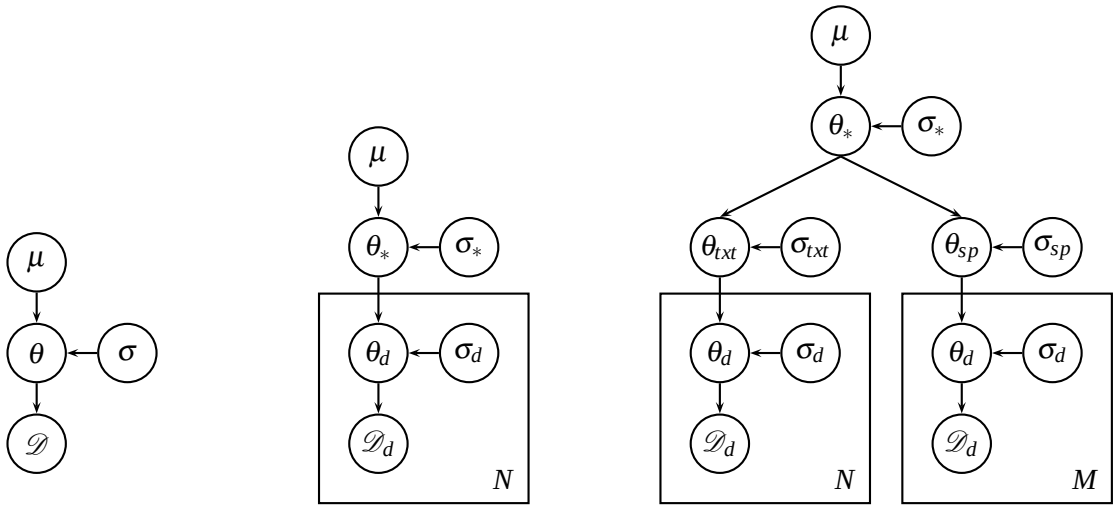
[1] This can be regarded as a Bayesian prior or as weight regularization; we adopt the former perspective here.

2.2 Formal Model

Our domain adaptation model is based on a hierarchical Bayesian prior, through which the domain-specific parameters are tied. The model is very general-purpose, and can be applied to any discriminative learning task for which one would typically put a prior with a mean over the parameters. We will build up to it by first describing a general, single-domain, discriminative learning task, and then we will show how to modify this model to construct our hierarchical Bayesian domain adaptation model. In a typical discriminative probabilistic model, the learning process consists of optimizing the log conditional likelihood of the data with respect to the parameters, $\mathscr{L}_{orig}(\mathscr{D};\theta)$. This likelihood function can take on many forms: logistic regression, a conditional Markov model, a conditional random field, as well as others. It is common practice to put a zero-mean Gaussian prior over the parameters, leading to the following objective, for which we wish to find the optimal parameter values:

$$\arg\max_{\theta} \left(\mathscr{L}_{orig}(\mathscr{D};\theta) - \sum_i \frac{\theta_i^2}{2\sigma^2} \right) \qquad (1)$$

From a graphical models perspective, this looks like Figure 1(a), where μ is the mean for the prior (in our case, zero), σ^2 is the variance for the prior, θ are the parameters, or feature weights, and $\mathscr{D}$ is the data. Now we will extend this single-domain model into a multi-domain model (illustrated in Figure 1(b)). Each feature weight θ_i is replicated once for each domain, as well as for a top-level set of parameters. We will refer to the parameters for domain d as θ_d, with individual components $\theta_{d,i}$, the top-level parameters as θ_*, and all parameters collectively as θ. All of the power of our model stems from the relationship between these sets of param-



(a) **(b)** **(c)**

Figure 1: **(a)** No domain adaptation. The model parameters, θ, are normally distributed, with mean μ (typically zero) and variance σ^2. The likelihood of the data, $\mathscr{D}$, is dependent on the model parameters. The form of the data distribution depends on the underlying model (e.g., logistic regression, or a CRF). **(b)** Our hierarchical domain adaptation model. The top-level parameters, θ_*, are normally distributed, with mean μ (typically zero) and variance σ_*^2. There is a plate for each domain. Within each plate, the domain-specific parameters, θ_d are normally distributed, with mean θ_* and variance σ_d^2. **(c)** Our hierarchical domain adaptation model, with an extra level of structure. In this example, the domains are further split into text and speech super-domains, each of which has its own set of parameters (θ_{txt} and σ_{txt} for text and θ_{sp} and σ_{sp} for speech). θ_d is normally distributed with mean θ_{txt} if domain d is in the text super-domain, and θ_{sp} if it is in the speech super-domain.

eters. First, we place a zero-mean Gaussian prior over the top level parameters θ_*. Then, these top level parameters are used as the mean for a Gaussian prior placed over each of the domain-specific parameters θ_d. These domain-specific parameters are then the parameters used in the original conditional log likelihood functions for each domain. The domain-specific parameter values jointly influence an appropriate value for the higher-level parameters. Conversely, the higher-level parameters will largely determine the domain-specific parameters when there is little or no evidence from within a domain, but can be overriden by domain-specific evidence when it clearly goes against the general picture (for instance *Leeds* is normally a *location*, but within the *sports* domain is usually an *organization* (football team)).

The beauty of this model is that the degree of influence each domain exerts over the others, for each parameter, is based on the amount of evidence each domain has about that parameter. If a domain has a lot of evidence for a feature weight, then that evidence will outweigh the effect of the prior. However, when a domain lacks evidence for a parameter the opposite occurs, and the prior (whose value is determined by evidence in the other domains) will have a greater effect on the parameter value.

To achieve this, we modify the objective function. We now sum over the log likelihood for all domains, including a Gaussian prior for each domain, but which is now centered around θ_*, the top-level parameters. Outside of this summation, we have a Gaussian prior over the top-level parameters which is identical to the prior in the original model:

$$\mathscr{L}_{hier}(\mathscr{D};\theta) = \sum_d \left(\mathscr{L}_{orig}(\mathscr{D}_d;\theta_d) - \sum_i \frac{(\theta_{d,i}-\theta_{*,i})^2}{2\sigma_d^2} \right) - \sum_i \frac{(\theta_{*,i})^2}{2\sigma_*^2} \quad (2)$$

where σ_d^2 and σ_*^2 are variances on the priors over the parameters for all the domains, as well as the top-level parameters. The graphical models representation is shown in Figure 1(b).

One potential source of confusion is with respect to the directed or undirected nature of our domain adaptation model, and the underlying model of the data. Our hierarchical Bayesian domain adaptation model is *directed*, as illustrated in Figure 1. However, somewhat counterintuitively, the underlying (original) model of the data can be either *directed* or *undirected*, and for our experiments we use undi-

rected, conditional random field-based models. The directed domain adaptation model can be viewed as a model of the parameters, and those parameter weights are used by the underlying data model. In Figure 1, the entire data model is represented by a single node, $\mathscr{D}$, conditioned on the parameters, θ or θ_d. The form of that model can then be almost anything, including an undirected model.

From an implementation perspective, the objective function is not much more difficult to implement than the original single-domain model. For all of our experiments, we optimized the log likelihood using L-BFGS, which requires the function value and partial derivatives of each parameter. The new partial derivatives for the domain-specific parameters (but not the top-level parameters) utilize the same partial derivatives as in the original model. The only change in the calculations is with respect to the priors. The partial derivatives for the domain-specific parameters are:

$$\frac{\partial \mathscr{L}_{hier}(\mathscr{D};\theta)}{\partial \theta_{d,i}} = \frac{\partial \mathscr{L}_d(\mathscr{D}_d,\theta_d)}{\partial \theta_{d,i}} - \frac{\theta_{d,i}-\theta_{*,i}}{\sigma_d^2} \quad (3)$$

and the derivatives for the top level parameters θ_* are:

$$\frac{\partial \mathscr{L}_{hier}(\mathscr{D};\theta)}{\partial \theta_{*,i}} = \left(\sum_d \frac{\theta_{*,i}-\theta_{d,i}}{\sigma_d^2}\right) - \frac{\theta_{*,i}}{\sigma_*^2} \quad (4)$$

This function is convex. Once the optimal parameters have been learned, the top level parameters can be discarded, since the runtime model for each domain is the same as the original (single-domain) model, parameterized by the parameters learned for that domain in the hierarchical model. However, it may be useful to retain the top-level parameters for use in adaptation to further domains in the future.

In our model there are d extra hyper-parameters which can be tuned. These are the variances σ_d^2 for each domain. When this value is large then the prior has little influence, and when set high enough will be equivalent to training each model separately. When this value is close to zero the prior has a strong influence, and when it is sufficiently close to zero then it will be equivalent to completely tying the parameters, such that $\theta_{d_1,i} = \theta_{d_2,i}$ for all domains. Despite having many more parameters, for both of the tasks on which we performed experiments, we found that our model did not take much more time to train that a baseline model trained on all of the data concatenated together.

2.3 Model Generalization

The model as presented thus far can be viewed as a two level tree, with the top-level parameters at the root, and the domain-specific ones at the leaves. However, it is straightforward to generalize the model to any tree structure. In the generalized version, the domain-specific parameters would still be at the leaves, the top-level parameters at the root, but new mid-level parameters can be added based on beliefs about how similar the various domains are. For instance, if one had four datasets, two of which contained speech data and two of which contained newswire, then it might be sensible to have two sets of mid-level parameters, one for the speech data and one for the newswire data, as illustrated in Figure 1(c). This would allow the speech domains to influence one another more than the newswire domains, and vice versa.

2.4 Formalization of (Daumé III, 2007)

As mentioned earlier, our model is equivalent to that presented in (Daumé III, 2007), and can be viewed as a formal version of his model.[2] In his presentation, the adapation is done through feature augmentation. Specifically, for each feature in the original version, a new version is created for each domain, as well as a general, domain-independent version of the feature. For each datum, two versions of each original feature are present: the version for that datum's domain, and the domain independent one.

The equivalence between the two models can be shown with simple arithmetic. Recall that the log likelihood of our model is:

$$\sum_d \left(\mathscr{L}_{orig}(\mathscr{D}_d;\theta_d) - \sum_i \frac{(\theta_{d,i}-\theta_{*,i})^2}{2\sigma_d^2} \right) - \sum_i \frac{(\theta_{*,i})^2}{2\sigma_*^2}$$

We now introduce a new variable $\psi_d = \theta_d - \theta_*$, and plug it into the equation for log likelihood:

$$\sum_d \left(\mathscr{L}_{orig}(\mathscr{D}_d;\psi_d+\theta_*) - \sum_i \frac{(\psi_{d,i})^2}{2\sigma_d^2} \right) - \sum_i \frac{(\theta_{*,i})^2}{2\sigma_*^2}$$

The result is the model of (Daumé III, 2007), where the ψ_d are the domain-specific feature weights, and θ_d are the domain-independent feature weights. In his formulation, the variances $\sigma_d^2 = \sigma_*^2$ for all domains d.

This separation of the domain-specific and independent variances was critical to our improved performance. When using a Gaussian prior there are

[2] Many thanks to David Vickrey for pointing this out to us.

two parameters set by the user: the mean, μ (usually zero), and the variance, σ^2. Technically, each of these parameters is actually a vector, with an entry for each feature, but almost always the vectors are uniform and the same parameter is used for each feature (there are exceptions, e.g. (Lee et al., 2007)). Because Daumé III (2007) views the adaptation as merely augmenting the feature space, each of his features has the same prior mean and variance, regardless of whether it is domain specific or independent. He could have set these parameters differently, but he did not.[3] In our presentation of the model, we explicitly represent different variances for each domain, as well as the top level parameters. We found that specifying different values for the domain specific versus domain independent variances significantly improved performance, though we found no gains from using different values for the different domain specific variances. The values were set based on development data.

3 Named Entity Recognition

For our first set of experiments, we used a linear-chain, conditional random field (CRF) model, trained for named entity recognition (NER). The use of CRFs for sequence modeling has become standard so we will omit the model details; good explanations can be found in a number of places (Lafferty et al., 2001; Sutton and McCallum, 2007). Our features were based on those in (Finkel et al., 2005).

3.1 Data

We used three named entity datasets, from the CoNLL 2003, MUC-6 and MUC-7 shared tasks. CoNLL is British newswire, while MUC-6 and MUC-7 are both American newswire. Arguably MUC-6 and MUC-7 should not count as separate domains, but because they were annotated separately, for different shared tasks, we chose to treat them as such, and feel that our experimental results justify the distinction. We used the standard train and test sets for each domain, which for CoNLL corresponds to the (more difficult) testb set. For details about the number of training and test words in each dataset, please see Table 1.

One interesting challenge in dealing with both CoNLL and MUC data is that the label sets differ.

	# Train Words	# Test Words
MUC-6	165,082	15,032
MUC-7	89,644	64,490
CoNLL	203,261	46,435

Table 1: Number of words in the training and test sets for each of the named entity recognition datasets.

CoNLL has four classes: *person*, *organization*, *location*, and *misc*. MUC data has seven classes: *person*, *organization*, *location*, *percent*, *date*, *time*, and *money*. They overlap in the three core classes (*person*, *organization*, and *location*), but CoNLL has one additional class and MUC has four additional classes.

The differences in the label sets led us to perform two sets of experiments for the baseline and hierarchical Bayesian models. In the first set of experiments, at training time, the model allows any label from the union of the label sets, regardless of whether that label was legal for the domain. At test time, we would ignore guesses made by the model which were inconsistent with the allowed labels for that domain.[4] In the second set of experiments, we restricted the model at training time to only allow legal labels for each domain. At test time, the domain was specified, and the model was once again restricted so that words would never be tagged with a label outside of that domain's label set.

3.2 Experimental Results and Discussion

In our experiments, we compared our model to several strong baselines, and the full set of results is in Table 2. The models we used were:

TARGET ONLY. Trained and tested on only the data for that domain.

ALL DATA. Trained and tested on data from all domains, concatenated into one large dataset.

ALL DATA*. Same as ALL DATA, but restricted possible labels for each word based on domain.

DAUME07. Trained and tested using the same technique as (Daumé III, 2007). We note that they present results using per-token label accuracy, while we used the more standard entity precision, recall, and F score (as in the CoNLL 2003 shared task).

[3] Although he alludes to the potential for something similar in the last section of his paper, when discussing the kernelization interpretation of his approach.

[4] We treated them identically to the background symbol. So, for instance, labelling a word a *date* in the CoNLL data had no effect on the score.

Named Entity Recognition

Model	Precision	Recall	F1
MUC-6			
TARGET ONLY	86.74	80.10	83.29
ALL DATA*	85.04	83.49	84.26
ALL DATA	86.00	82.71	84.32
DAUME07*	87.83	83.41	85.56
DAUME07	87.81	82.23	85.46
HIER BAYES*	88.59	84.97	86.74
HIER BAYES	**88.77**	**85.14**	**86.92**
MUC-7			
TARGET ONLY	81.17	70.23	75.30
ALL DATA*	81.66	76.17	78.82
ALL DATA	82.20	70.91	76.14
DAUME07*	**83.33**	75.42	79.18
DAUME07	83.51	75.63	79.37
HIER BAYES*	82.90	76.95	79.82
HIER BAYES	83.17	**77.02**	**79.98**
CoNLL			
TARGET ONLY	85.55	84.72	85.13
ALL DATA*	86.34	84.45	85.38
ALL DATA	86.58	83.90	85.22
DAUME07*	86.09	85.06	85.57
DAUME07	86.35	**85.26**	**85.80**
HIER BAYES*	86.33	85.06	85.69
HIER BAYES	**86.51**	85.13	**85.81**

Table 2: Named entity recognition results for each of the models. With the exception of the TARGET ONLY model, all three datasets were combined when training each of the models.

DAUME07*. Same as DAUME07, but restricted possible labels for each word based on domain.

HIER BAYES. Our hierarchical Bayesian domain adaptation model.

HIER BAYES*. Same as HIER BAYES, but restricted possible labels for each word based on the domain.

For all of the baseline models, and for the top level-parameters in the hierarchical Bayesian model, we used $\sigma = 1$. For the domain-specific parameters, we used $\sigma_d = 0.1$ for all domains.

The HIER BAYES model outperformed all baselines for both of the MUC datasets, and tied with the DAUME07 for CoNLL. The largest improvement was on MUC-6, where HIER BAYES outperformed DAUME07*, the second best model, by 1.36%. This improvement is greater than the improvement made by that model over the ALL DATA* baseline. To assess significance we used a document-level paired t-test (over all of the data combined), and found that HIER BAYES significantly outperformed all of the baselines (not including HIER BAYES*) with greater than 95% confidence.

For both the HIER BAYES and DAUME07 models, we found that performance was better for the variant which did not restrict possible labels based on the domain, while the ALL DATA model did benefit from the label restriction. For HIER BAYES and DAUME07, this result may be due to the structure of the models. Because both models have domain-specific features, the models likely learned that these labels were never actually allowed. However, when a feature does not occur in the data for a particular domain, then the domain-specific parameter for that feature will have positive weight due to evidence present in the other domains, which at test time can lead to assigning an illegal label to a word. This information that a word may be of some other (unknown to that domain) entity type may help prevent the model from mislabeling the word. For example, in CoNLL, nationalities, such as *Iraqi* and *American*, are labeled as *misc*. If a previously unseen nationality is encountered in the MUC testing data, the MUC model may be tempted to label is as a *location*, but this evidence from the CoNLL data may prevent that, by causing it to instead be labeled *misc*, a label which will subsequently be ignored.

In typical domain adaptation work, showing gains is made easier by the fact that the amount of training data in the *target* domain is comparatively small. Within the multi-task learning setting, it is more challenging to show gains over the ALL DATA baseline. Nevertheless, our results show that, so long as the amount of data in each domain is not widely disparate, it is possible to achieve gains on all of the domains simultaneously.

4 Dependency Parsing

4.1 Parsing Model

We also tested our model on an untyped dependency parsing task, to see how it performs on a more structurally complex task than sequence modeling. To our knowledge, the discriminatively trained dependency model we used has not been previously published, but it is very similar to recent work on discriminative constituency parsing (Finkel and Manning, 2008). Due to space restrictions, we cannot give a complete treatment of the model, but will give an overview.

We built a CRF-based model, optimizing the likelihood of the parse, conditioned on the words and parts of speech of the sentence. At the heart of our model is the Eisner dependency grammar chart-parsing algorithm (Eisner, 1996), which allows for efficient computation of inside and outside scores. The Eisner algorithm, originally designed for generative parsing, decomposes the probability of a dependency parse into the probabilities of each attachment of a dependent to its parent, and the probabilities of each parent stopping taking dependents. These probabilities can be conditioned on the child, parent, and direction of the dependency. We used a slight modification of the algorithm which allows each probability to also be conditioned on whether there is a previous dependent. While the unmodified version of the algorithm includes stopping probabilities, conditioned on the parent and direction, they have no impact on which parse for a particular sentence is most likely, because all words must eventually stop taking dependents. However, in the modified version, the stopping probability is also conditioned on whether or not there is a previous dependent, so this probability does make a difference.

While the Eisner algorithm computes locally normalized probabilities for each attachment decision, our model computes unnormalized scores. From a graphical models perspective, our parsing model is undirected, while the original model is directed.[5] The score for a particular tree decomposes the same way in our model as in the original Eisner model, but it is globally normalized instead of locally normalized. Using the inside and outside scores we can compute partial derivatives for the feature weights, as well as the value of the normalizing constant needed to determine the probability of a particular parse. This is done in a manner completely analogous to (Finkel and Manning, 2008). Partial derivatives and the function value are all that is needed to find the optimal feature weights using L-BFGS.[6]

Features are computed over each attachment and stopping decision, and can be conditioned on the parent, dependent (or none, if it is a stopping decision), direction of attachment, whether there is a previous dependent in that direction, and the words and parts of speech of the sentence. We used the same features as (McDonald et al., 2005), augmented with information about whether or not a dependent is the first dependent (information they did not have).

4.2 Data

For our dependency parsing experiments, we used LDC2008T04 OntoNotes Release 2.0 data (Hovy et al., 2006). This dataset is still in development, and includes data from seven different domains, labeled for a number of tasks, including PCFG trees. The domains span both newswire and speech from multiple sources. We converted the PCFG trees into dependency trees using the Collins head rules (Collins, 2003). We also omitted the WSJ portion of the data, because it follows a different annotation scheme from the other domains.[7] For each of the remaining six domains, we aimed for an 75/25 data split, but because we divided the data using the provided sections, this split was fairly rough. The number of training and test sentences for each domain are specified in the Table 3, along with our results.

4.3 Experimental Results and Discussion

We compared the same four domain adaptation models for dependency parsing as we did for the named entity experiments, once again setting $\sigma = 1.0$ and $\sigma_d = 0.1$. Unlike the named entity experiments however, there were no label set discrepencies between the domains, so only one version of each domain adaptation model was necessary, instead of the two versions in that section.

Our full dependency parsing results can be found in Table 3. Firstly, we found that DAUME07, which had outperformed the ALL DATA baseline for the sequence modeling task, performed worse than the

[5] The dependencies themselves are still *directed* in both cases, it is just the underlying graphical model used to compute the likelihood of a parse which changes from a directed model to an undirected model.

[6] In (Finkel and Manning, 2008) we used stochastic gradient descent to optimize our weights because our function evaluation was too slow to use L-BFGS. We did not encounter this problem in this setting.

[7] Specifically, all the other domains use the "new" Penn Treebank annotation style, whereas the WSJ data is still in the "traditional" annotation style, familiar from the past decade's work in Penn Treebank parsing. The major changes are in hyphenation and NP structure. In the new annotation style, many hyphenated words are separated into multiple tokens, with a new part-of-speech tag given to the hyphens, and leftward-branching structure inside noun phrases is indicated by use of a new NML phrasal category. The treatment of hyphenated words, in particular, makes the two annotation styles inconsistent, and so we could not work with all the data together.

Dependency Parsing								
	Training		Testing		TARGET	ALL		HIER
	Range	# Sent	Range	# Sent	ONLY	DATA	DAUME07	BAYES
ABC	0–55	1195	56–69	199	83.32%	**88.97%**	87.30%	88.68%
CNN	0–375	5092	376–437	1521	85.53%	87.09%	86.41%	**87.26%**
MNB	0–17	509	18–25	245	77.06%	86.41%	84.70%	**86.71%**
NBC	0–29	552	30–39	149	76.21%	**85.82%**	85.01%	85.32%
PRI	0–89	1707	90–112	394	87.65%	90.28%	89.52%	**90.59%**
VOA	0–198	1512	199–264	383	89.17%	**92.11%**	90.67%	**92.09%**

Table 3: Dependency parsing results for each of the domain adaptation models. Performance is measured as unlabeled attachment accuracy.

baseline here, indicating that the transfer of information between domains in the more structurally complicated task is inherently more difficult. Our model's gains over the ALL DATA baseline are quite small, but we tested their significance using a sentence-level paired t-test (over all of the data combined) and found them to be significant at $p < 10^{-5}$. We are unsure why some domains improved while others did not. It is not simply a consequence of training set size, but may be due to qualities of the domains themselves.

5 Related Work

We already discussed the relation of our work to (Daumé III, 2007) in Section 2.4. Another piece of similar work is (Chelba and Acero, 2004), who also modify their prior. Their work is limited to two domains, a source and a target, and their algorithm has a two stage process: First, train a classifier on the source data, and then use the learned weights from that classifier as the mean for a Gaussian prior when training a new model on just the target data.

Daumé III and Marcu (2006) also took a Bayesian approach to domain adaptation, but structured their model in a very different way. In their model, it is assumed that each datum within a domain is either a domain-specific datum, or a general datum, and then domain-specific and general weights were learned. Whether each datum is domain-specific or general is not known, so they developed an EM based algorithm for determining this information while simultaneously learning the feature weights. Their model had good performance, but came with a 10 to 15 times slowdown at training time. Our slowest dependency parser took four days to train, making this model close to infeasible for learning on that data.

Outside of the NLP community there has been much similar work making use of hierarchical Bayesian priors to tie parameters across multiple, similar tasks. Evgeniou et al. (2005) present a similar model, but based on support vector machines, to predict the exam scores of students. Elidan et al. (2008) make us of an *undirected* Bayesian transfer hierarchy to jointly model the shapes of different mammals. The complete literature on related multi-task learning is too large to fully discuss here, but we direct the reader to (Baxter, 1997; Caruana, 1997; Yu et al., 2005; Xue et al., 2007). For a more general discussion of hierarchical priors, we recommend Chapter 5 of (Gelman et al., 2003) and Chapter 12 of (Gelman and Hill, 2006).

6 Conclusion and Future Work

In this paper we presented a new model for domain adaptation, based on a hierarchical Bayesian prior, which allows information to be shared between domains when information is sparse, while still allowing the data from a particular domain to override the information from other domains when there is sufficient evidence. We outperformed previous work on a sequence modeling task, and showed improvements on dependency parsing, a structurally more complex problem, where previous work failed. Our model is practically useful and does not require significantly more time to train than a baseline model using the same data (though it does require more memory, proportional to the number of domains). In the future we would like to see if the model could be adapted to improve performance on data from a new domain, potentially by using the top-level weights which should be less domain-dependent.

Acknowledgements

The first author is supported by a Stanford Graduate Fellowship. We also thank David Vickrey for his helpful comments and observations.

References

J. Baxter. 1997. A bayesian/information theoretic model of learning to learn via multiple task sampling. In *Machine Learning*, volume 28.

R. Caruana. 1997. Multitask learning. In *Machine Learning*, volume 28.

Ciprian Chelba and Alex Acero. 2004. Adaptation of a maximum entropy capitalizer: Little data can help a lot. In *EMNLP 2004*.

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

Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, Copenhagen.

Gal Elidan, Benjamin Packer, Geremy Heitz, and Daphne Koller. 2008. Convex point estimation using undirected bayesian transfer hierarchies. In *UAI 2008*.

T. Evgeniou, C. Micchelli, and M. Pontil. 2005. Learning multiple tasks with kernel methods. In *Journal of Machine Learning Research*.

Jenny Rose Finkel and Christopher D. Manning. 2008. Efficient, feature-based conditional random field parsing. In *ACL/HLT-2008*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005*.

Andrew Gelman and Jennifer Hill. 2006. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

A. Gelman, J. B. Carlin, H. S. Stern, and Donald D. B. Rubin. 2003. *Bayesian Data Analysis*. Chapman & Hall.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *HLT-NAACL 2006*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*.

Su-In Lee, Vassil Chatalbashev, David Vickrey, and Daphne Koller. 2007. Learning a meta-level prior for feature relevance from multiple related tasks. In *ICML '07: Proceedings of the 24th international conference on Machine learning*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *ACL 2005*.

Charles Sutton and Andrew McCallum. 2007. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. 2007. Multi-task learning for classification with dirichlet process priors. *J. Mach. Learn. Res.*, 8.

Kai Yu, Volker Tresp, and Anton Schwaighofer. 2005. Learning gaussian processes from multiple tasks. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*.

Online EM for Unsupervised Models

Percy Liang **Dan Klein**
Computer Science Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720
{pliang,klein}@cs.berkeley.edu

Abstract

The (batch) EM algorithm plays an important role in unsupervised induction, but it sometimes suffers from slow convergence. In this paper, we show that online variants (1) provide significant speedups and (2) can even find *better* solutions than those found by batch EM. We support these findings on four unsupervised tasks: part-of-speech tagging, document classification, word segmentation, and word alignment.

1 Introduction

In unsupervised NLP tasks such as tagging, parsing, and alignment, one wishes to induce latent linguistic structures from raw text. Probabilistic modeling has emerged as a dominant paradigm for these problems, and the EM algorithm has been a driving force for learning models in a simple and intuitive manner.

However, on some tasks, EM can converge slowly. For instance, on unsupervised part-of-speech tagging, EM requires over 100 iterations to reach its peak performance on the Wall-Street Journal (Johnson, 2007). The slowness of EM is mainly due to its batch nature: Parameters are updated only once after each pass through the data. When parameter estimates are still rough or if there is high redundancy in the data, computing statistics on the entire dataset just to make one update can be wasteful.

In this paper, we investigate two flavors of online EM—*incremental EM* (Neal and Hinton, 1998) and *stepwise EM* (Sato and Ishii, 2000; Cappé and Moulines, 2009), both of which involve updating parameters after each example or after a mini-batch (subset) of examples. Online algorithms have the potential to speed up learning by making updates more frequently. However, these updates can be seen as noisy approximations to the full batch update, and this noise can in fact impede learning.

This tradeoff between speed and stability is familiar to online algorithms for convex supervised learning problems—e.g., Perceptron, MIRA, stochastic gradient, etc. Unsupervised learning raises two additional issues: (1) Since the EM objective is non-convex, we often get convergence to different local optima of varying quality; and (2) we evaluate on accuracy metrics which are at best loosely correlated with the EM likelihood objective (Liang and Klein, 2008). We will see that these issues can lead to surprising results.

In Section 4, we present a thorough investigation of online EM, mostly focusing on stepwise EM since it dominates incremental EM. For stepwise EM, we find that choosing a good stepsize and mini-batch size is important but can fortunately be done adequately without supervision. With a proper choice, stepwise EM reaches the same performance as batch EM, but much more quickly. Moreover, it can even surpass the performance of batch EM. Our results are particularly striking on part-of-speech tagging: Batch EM crawls to an accuracy of 57.3% after 100 iterations, whereas stepwise EM shoots up to 65.4% after just two iterations.

2 Tasks, models, and datasets

In this paper, we focus on unsupervised induction via probabilistic modeling. In particular, we define a probabilistic model $p(\mathbf{x}, \mathbf{z}; \theta)$ of the input $\mathbf{x}$ (e.g.,

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 611–619,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

a sentence) and hidden output $\mathbf{z}$ (e.g., a parse tree) with parameters θ (e.g., rule probabilities). Given a set of unlabeled examples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, the standard training objective is to maximize the marginal log-likelihood of these examples:

$$\ell(\theta) = \sum_{i=1}^{n} \log p(\mathbf{x}^{(i)}; \theta). \quad (1)$$

A trained model $\hat{\theta}$ is then evaluated on the accuracy of its predictions: $\text{argmax}_{\mathbf{z}}\, p(\mathbf{z} \mid \mathbf{x}^{(i)}; \hat{\theta})$ against the true output $\mathbf{z}^{(i)}$; the exact evaluation metric depends on the task. What makes unsupervised induction hard at best and ill-defined at worst is that the training objective (1) does not depend on the true outputs at all.

We ran experiments on four tasks described below. Two of these tasks—part-of-speech tagging and document classification—are "clustering" tasks. For these, the output $\mathbf{z}$ consists of labels; for evaluation, we map each predicted label to the true label that maximizes accuracy. The other two tasks—segmentation and alignment—only involve unlabeled combinatorial structures, which can be evaluated directly.

Part-of-speech tagging For each sentence $\mathbf{x} = (x_1, \ldots, x_\ell)$, represented as a sequence of words, we wish to predict the corresponding sequence of part-of-speech (POS) tags $\mathbf{z} = (z_1, \ldots, z_\ell)$. We used a simple bigram HMM trained on the Wall Street Journal (WSJ) portion of the Penn Treebank (49208 sentences, 45 tags). No tagging dictionary was used. We evaluated using per-position accuracy.

Document classification For each document $\mathbf{x} = (x_1, \ldots, x_\ell)$ consisting of ℓ words,[1] we wish to predict the document class $\mathbf{z} \in \{1, \ldots, 20\}$. Each document $\mathbf{x}$ is modeled as a bag of words drawn independently given the class $\mathbf{z}$. We used the 20 Newsgroups dataset (18828 documents, 20 classes). We evaluated on class accuracy.

Word segmentation For each sentence $\mathbf{x} = (x_1, \ldots, x_\ell)$, represented as a sequence of English phonemes or Chinese characters without spaces separating the words, we would like to predict a segmentation of the sequence into words $\mathbf{z} = (z_1, \ldots, z_{|\mathbf{z}|})$, where each segment (word) z_i is a contiguous subsequence of $1, \ldots, \ell$. Since the naïve unigram model has a degenerate maximum likelihood solution that makes each sentence a separate word, we incorporate a penalty for longer segments: $p(\mathbf{x}, \mathbf{z}; \theta) \propto \prod_{k=1}^{|\mathbf{z}|} p(\mathbf{x}_{z_k}; \theta) e^{-|z_k|^\beta}$, where $\beta > 1$ determines the strength of the penalty. For English, we used $\beta = 1.6$; Chinese, $\beta = 2.5$. To speed up inference, we restricted the maximum segment length to 10 for English and 5 for Chinese.

We applied this model on the Bernstein-Ratner corpus from the CHILDES database used in Goldwater et al. (2006) (9790 sentences) and the Academia Sinica (AS) corpus from the first SIGHAN Chinese word segmentation bakeoff (we used the first 100K sentences). We evaluated using F_1 on word tokens.

To the best of our knowledge, our penalized unigram model is new and actually beats the more complicated model of Johnson (2008) 83.5% to 78%, which had been the best published result on this task.

Word alignment For each pair of translated sentences $\mathbf{x} = (e_1, \ldots, e_{n_e}, f_1, \ldots, f_{n_f})$, we wish to predict the word alignments $\mathbf{z} \in \{0, 1\}^{n_e n_f}$. We trained two IBM model 1s using agreement-based learning (Liang et al., 2008). We used the first 30K sentence pairs of the English-French Hansards data from the NAACL 2003 Shared Task, 447+37 of which were hand-aligned (Och and Ney, 2003). We evaluated using the standard alignment error rate (AER).

3 EM algorithms

Given a probabilistic model $p(\mathbf{x}, \mathbf{z}; \theta)$ and unlabeled examples $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$, recall we would like to maximize the marginal likelihood of the data (1). Let $\phi(\mathbf{x}, \mathbf{z})$ denote a mapping from a fully-labeled example $(\mathbf{x}, \mathbf{z})$ to a vector of sufficient statistics (counts in the case of multinomials) for the model. For example, one component of this vector for HMMs would be the number of times state 7 emits the word "house" in sentence $\mathbf{x}$ with state sequence $\mathbf{z}$. Given a vector of sufficient statistics μ, let $\theta(\mu)$ denote the maximum likelihood estimate. In our case, $\theta(\mu)$ are simply probabilities obtained by normalizing each block of counts. This closed-form

[1] We removed the 50 most common words and words that occurred fewer than 5 times.

Batch EM

$\mu \leftarrow$ initialization
for each iteration $t = 1, \ldots, T$:
  $\mu' \leftarrow 0$
  for each example $i = 1, \ldots, n$:
    $s'_i \leftarrow \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{x}^{(i)}; \theta(\mu))\, \phi(\mathbf{x}^{(i)}, \mathbf{z})$ [inference]
    $\mu' \leftarrow \mu' + s'_i$ [accumulate new]
  $\mu \leftarrow \mu'$ [replace old with new]

Incremental EM (iEM)

$s_i \leftarrow$ initialization for $i = 1, \ldots, n$
$\mu \leftarrow \sum_{i=1}^{n} s_i$
for each iteration $t = 1, \ldots, T$:
  for each example $i = 1, \ldots, n$ in random order:
    $s'_i \leftarrow \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{x}^{(i)}; \theta(\mu))\, \phi(\mathbf{x}^{(i)}, \mathbf{z})$ [inference]
    $\mu \leftarrow \mu + s'_i - s_i$; $s_i \leftarrow s'_i$ [replace old with new]

Stepwise EM (sEM)

$\mu \leftarrow$ initialization; $k = 0$
for each iteration $t = 1, \ldots, T$:
  for each example $i = 1, \ldots, n$ in random order:
    $s'_i \leftarrow \sum_{\mathbf{z}} p(\mathbf{z} \mid \mathbf{x}^{(i)}; \theta(\mu))\, \phi(\mathbf{x}^{(i)}, \mathbf{z})$ [inference]
    $\mu \leftarrow (1-\eta_k)\mu + \eta_k s'_i$; $k \leftarrow k+1$ [towards new]

solution is one of the features that makes EM (both batch and online) attractive.

3.1 Batch EM

In the (batch) EM algorithm, we alternate between the E-step and the M-step. In the E-step, we compute the expected sufficient statistics μ' across all the examples based on the posterior over $\mathbf{z}$ under the current parameters $\theta(\mu)$. In all our models, this step can be done via a dynamic program (for example, forward-backward for POS tagging).

In the M-step, we use these sufficient statistics μ' to re-estimate the parameters. Since the M-step is trivial, we represent it implicitly by $\theta(\cdot)$ in order to concentrate on the computation of the sufficient statistics. This focus will be important for online EM, so writing batch EM in this way accentuates the parallel between batch and online.

3.2 Online EM

To obtain an online EM algorithm, we store a single set of sufficient statistics μ and update it after processing each example. For the i-th example, we compute sufficient statistics s'_i. There are two main variants of online EM algorithms which differ in exactly how the new s'_i is incorporated into μ.

The first is *incremental EM* (iEM) (Neal and Hinton, 1998), in which we not only keep track of μ but also the sufficient statistics $s_1, \ldots, s_n$ for each example ($\mu = \sum_{i=1}^{n} s_i$). When we process example i, we subtract out the old s_i and add the new s'_i.

Sato and Ishii (2000) developed another variant, later generalized by Cappé and Moulines (2009), which we call *stepwise EM* (sEM). In sEM, we interpolate between μ and s'_i based on a stepsize η_k (k is the number of updates made to μ so far).

The two algorithms are motivated in different ways. Recall that the log-likelihood can be lower bounded as follows (Neal and Hinton, 1998):

$$\ell(\theta) \geq \mathcal{L}(q_1, \ldots, q_n, \theta) \tag{2}$$
$$\stackrel{\text{def}}{=} \sum_{i=1}^{n} \Big[\sum_{\mathbf{z}} q_i(\mathbf{z} \mid \mathbf{x}^{(i)}) \log p(\mathbf{x}^{(i)}, \mathbf{z}; \theta) + H(q_i) \Big],$$

where $H(q_i)$ is the entropy of the distribution $q_i(\mathbf{z} \mid \mathbf{x}^{(i)})$. Batch EM alternates between optimizing $\mathcal{L}$ with respect to $q_1, \ldots, q_n$ in the E-step (represented implicitly via sufficient statistics μ') and with respect to θ in the M-step. Incremental EM alternates between optimizing with respect to a single q_i and θ.

Stepwise EM is motivated from the stochastic approximation literature, where we think of approximating the update μ' in batch EM with a single sample s'_i. Since one sample is a bad approximation, we interpolate between s'_i and the current μ. Thus, sEM can be seen as stochastic gradient in the space of sufficient statistics.

Stepsize reduction power α Stepwise EM leaves open the choice of the stepsize η_k. Standard results from the stochastic approximation literature state that $\sum_{k=0}^{\infty} \eta_k = \infty$ and $\sum_{k=0}^{\infty} \eta_k^2 < \infty$ are sufficient to guarantee convergence to a local optimum. In particular, if we take $\eta_k = (k+2)^{-\alpha}$, then any $0.5 < \alpha \leq 1$ is valid. The smaller the α, the larger the updates, and the more quickly we forget (decay) our old sufficient statistics. This can lead to swift progress but also generates instability.

Mini-batch size m We can add some stability to sEM by updating on multiple examples at once

instead of just one. In particular, partition the n examples into mini-batches of size m and run sEM, treating each mini-batch as a single example. Formally, for each $i = 0, m, 2m, 3m, \ldots$, first compute the sufficient statistics $s'_{i+1}, \ldots, s'_{i+m}$ on $\mathbf{x}^{(i+1)}, \ldots, \mathbf{x}^{(i+m)}$ and then update μ using $s'_{i+1} + \cdots + s'_{i+m}$. The larger the m, the less frequent the updates, but the more stable they are. In this way, mini-batches interpolate between a pure online ($m = 1$) and a pure batch ($m = n$) algorithm.[2]

Fast implementation Due to sparsity in NLP, the sufficient statistics of an example s'_i are nonzero for a small fraction of its components. For iEM, the time required to update μ with s'_i depends only on the number of nonzero components of s'_i. However, the sEM update is $\mu \leftarrow (1-\eta_k)\mu + \eta_k s'_i$, and a naïve implementation would take time proportional to the total number of components. The key to a more efficient solution is to note that $\theta(\mu)$ is invariant to scaling of μ. Therefore, we can store $S = \frac{\mu}{\prod_{j<k}(1-\eta_j)}$ instead of μ and make the following *sparse* update: $S \leftarrow S + \frac{\eta_k}{\prod_{j<k}(1-\eta_j)} s'_i$, taking comfort in the fact that $\theta(\mu) = \theta(S)$.

For both iEM and sEM, we also need to efficiently compute $\theta(\mu)$. We can do this by maintaining the normalizer for each multinomial block (sum of the components in the block). This extra maintenance only doubles the number of updates we have to make but allows us to fetch any component of $\theta(\mu)$ in constant time by dividing out the normalizer.

3.3 Incremental versus stepwise EM

Incremental EM increases $\mathcal{L}$ monotonically after each update by virtue of doing coordinate-wise ascent and thus is guaranteed to converge to a local optimum of both $\mathcal{L}$ and ℓ (Neal and Hinton, 1998). However, ℓ is not guaranteed to increase after each update. Stepwise EM might not increase either $\mathcal{L}$ or ℓ after each update, but it is guaranteed to converge to a local optimum of ℓ given suitable conditions on the stepsize discussed earlier.

Incremental and stepwise EM actually coincide under the following setting (Cappé and Moulines, 2009): If we set $(\alpha, m) = (1, 1)$ for sEM and initialize all $s_i = 0$ for iEM, then both algorithms make the same updates on the first pass through the data. They diverge thereafter as iEM subtracts out old s_is, while sEM does not even remember them.

One weakness of iEM is that its memory requirements grow linearly with the number of examples due to storing $s_1, \ldots, s_n$. For large datasets, these s_is might not even fit in memory, and resorting to physical disk would be very slow. In contrast, the memory usage of sEM does not depend on n.

The relationship between iEM and sEM (with $m = 1$) is analogous to the one between exponentiated gradient (Collins et al., 2008) and stochastic gradient for supervised learning of log-linear models. The former maintains the sufficient statistics of each example and subtracts out old ones whereas the latter does not. In the supervised case, the added stability of exponentiated gradient tends to yield better performance. For the unsupervised case, we will see empirically that remembering the old sufficient statistics offers no benefit, and much better performance can be obtained by properly setting (α, m) for sEM (Section 4).

4 Experiments

We now present our empirical results for batch EM and online EM (iEM and sEM) on the four tasks described in Section 2: part-of-speech tagging, document classification, word segmentation (English and Chinese), and word alignment.

We used the following protocol for all experiments: We initialized the parameters to a neutral setting plus noise to break symmetries.[3] Training was performed for 20 iterations.[4] No parameter smoothing was used. All runs used a fixed random seed for initializing the parameters and permuting the examples at the beginning of each iteration. We report two performance metrics: log-likelihood normalized by the number of examples and the task-specific accuracy metric (see Section 2). All numbers are taken from the final iteration.

[2] Note that running sEM with $m = n$ is similar but not equivalent to batch EM since old sufficient statistics are still interpolated rather than replaced.

[3] Specifically, for each block of multinomial probabilities $\theta_1, \ldots, \theta_K$, we set $\theta_k \propto \exp\{10^{-3}(1 + a_k)\}$, where $a_k \sim U[0, 1]$. Exception: for batch EM on POS tagging, we used 1 instead of 10^{-3}; more noise worked better.

[4] Exception: for batch EM on POS tagging, 100 iterations was needed to get satisfactory performance.

Stepwise EM (sEM) requires setting two optimization parameters: the stepsize reduction power α and the mini-batch size m (see Section 3.2). As Section 4.3 will show, these two parameters can have a large impact on performance. As a default rule of thumb, we chose $(\alpha, m) \in \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\} \times \{1, 3, 10, 30, 100, 300, 1\text{K}, 3\text{K}, 10\text{K}\}$ to maximize log-likelihood; let sEM$_\ell$ denote stepwise EM with this setting. Note that this setting requires no labeled data. We will also consider fixing $(\alpha, m) = (1, 1)$ (sEM$_i$) and choosing (α, m) to maximize accuracy (sEM$_a$).

In the results to follow, we first demonstrate that online EM is faster (Section 4.1) and sometimes leads to higher accuracies (Section 4.2). Next, we explore the effect of the optimization parameters (α, m) (Section 4.3), briefly revisiting the connection between incremental and stepwise EM. Finally, we show the stability of our results under different random seeds (Section 4.4).

4.1 Speed

One of the principal motivations for online EM is speed, and indeed we found this motivation to be empirically well-justified. Figure 1 shows that, across all five datasets, sEM$_\ell$ converges to a solution with at least comparable log-likelihood and accuracy with respect to batch EM, but sEM$_\ell$ does it anywhere from about 2 (word alignment) to 10 (POS tagging) times faster. This supports our intuition that more frequent updates lead to faster convergence. At the same time, note that the other two online EM variants in Figure 1 (iEM and sEM$_i$) are prone to catastrophic failure. See Section 4.3 for further discussion on this issue.

4.2 Performance

It is fortunate but perhaps not surprising that stepwise EM is faster than batch EM. But Figure 1 also shows that, somewhat surprisingly, sEM$_\ell$ can actually converge to a solution with higher accuracy, in particular on POS tagging and document classification. To further explore the accuracy-increasing potential of sEM, consider choosing (α, m) to maximize accuracy (sEM$_a$). Unlike sEM$_\ell$, sEM$_a$ does require labeled data. In practice, (α, m) can be tuned on a small labeled set along with any model hyperparameters.

	EM	sEM$_\ell$	sEM$_a$	α_ℓ	m_ℓ	α_a	m_a
POS	57.3	59.6	**66.7**	0.7	3	0.5	3
DOC	39.1	47.8	**49.9**	0.8	1K	0.5	3K
SEG(en)	80.5	80.7	**83.5**	0.7	1K	1.0	100
SEG(ch)	**78.2**	77.2	78.1	0.6	10K	1.0	10K
ALIGN	78.8	78.9	**78.9**	0.7	10K	0.7	10K

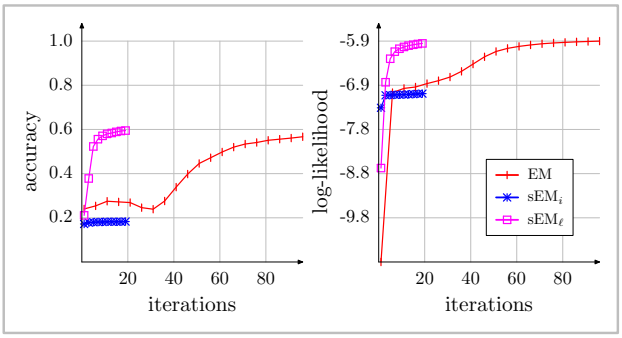
Table 1: Accuracy of batch EM and stepwise EM, where the optimization parameters (α, m) are tuned to either maximize log-likelihood (sEM$_\ell$) or accuracy (sEM$_a$). With an appropriate setting of (α, m), stepwise EM outperforms batch EM significantly on POS tagging and document classification.

Table 1 shows that sEM$_a$ improves the accuracy compared to batch EM even more than sEM$_\ell$. The result for POS is most vivid: After one iteration of batch EM, the accuracy is only at 24.0% whereas sEM$_a$ is already at 54.5%, and after two iterations, at 65.4%. Not only is this orders of magnitude faster than batch EM, batch EM only reaches 57.3% after 100 iterations.

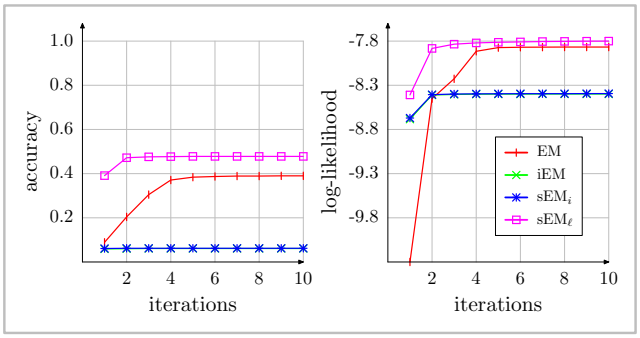
We get a similarly striking result for document classification, but the results for word segmentation and word alignment are more modest. A full understanding of this phenomenon is left as an open problem, but we will comment on one difference between the tasks where sEM improves accuracy and the tasks where it doesn't. The former are "clustering" tasks (POS tagging and document classification), while the latter are "structural" tasks (word segmentation and word alignment). Learning of clustering models centers around probabilities over words given a latent cluster label, whereas in structural models, there are no cluster labels, and it is the combinatorial structure (the segmentations and alignments) that drives the learning.

Likelihood versus accuracy From Figure 1, we see that stepwise EM (sEM$_\ell$) can outperform batch EM in both likelihood and accuracy. This suggests that stepwise EM is better at avoiding local minima, perhaps leveraging its stochasticity to its advantage.

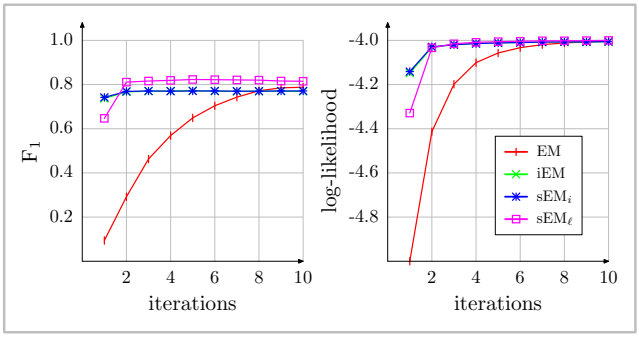
However, on POS tagging, tuning sEM to maximize accuracy (sEM$_a$) results in a slower increase in likelihood: compare sEM$_a$ in Figure 2 with sEM$_\ell$ in Figure 1(a). This shouldn't surprise us too much given that likelihood and accuracy are only loosely



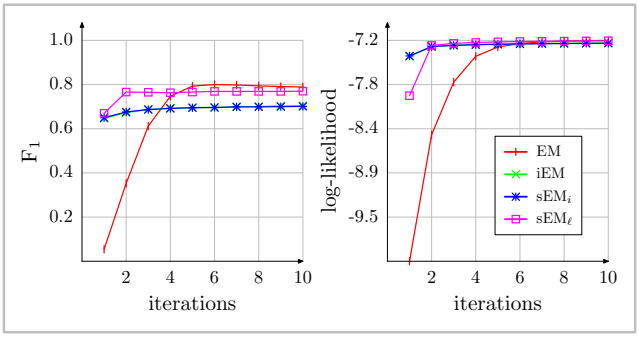
(a) POS tagging



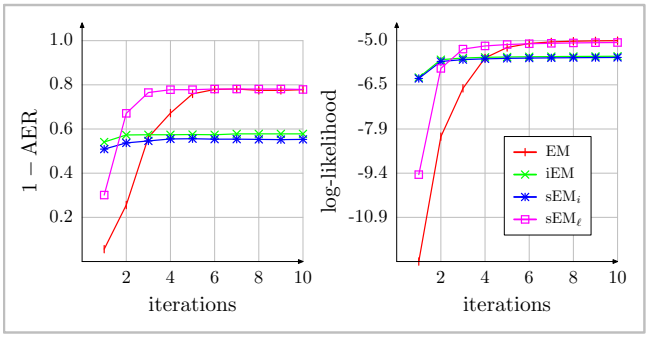
(b) Document classification



(c) Word segmentation (English)



(d) Word segmentation (Chinese)



(e) Word alignment

	accuracy		log-likelihood	
	EM	sEM$_\ell$	EM	sEM$_\ell$
POS	57.3	59.6	-6.03	-6.08
DOC	39.1	47.8	-7.96	-7.88
SEG(en)	80.5	80.7	-4.11	-4.11
SEG(ch)	78.2	77.2	-7.27	-7.28
ALIGN	78.8	78.9	-5.05	-5.12

(f) Results after convergence

Figure 1: Accuracy and log-likelihood plots for batch EM, incremental EM, and stepwise EM across all five datasets. sEM$_\ell$ outperforms batch EM in terms of convergence speed and even accuracy and likelihood; iEM and sEM$_i$ fail in some cases. We did not run iEM on POS tagging due to memory limitations; we expect the performance would be similar to sEM$_i$, which is not very encouraging (Section 4.3).

correlated (Liang and Klein, 2008). But it does suggest that stepwise EM is injecting a bias that favors accuracy over likelihood—a bias not at all reflected in the training objective.

We can create a hybrid (sEM$_a$+EM) that combines the strengths of both sEM$_a$ and EM: First run sEM$_a$ for 5 iterations, which quickly takes us to a part of the parameter space yielding good accuracies; then run EM, which quickly improves the likelihood. Fortunately, accuracy does not degrade as likelihood increases (Figure 2).

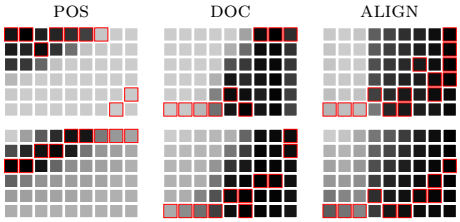
4.3 Varying the optimization parameters

Recall that stepwise EM requires setting two optimization parameters: the stepsize reduction power α and the mini-batch size m. We now explore the effect of (α, m) on likelihood and accuracy.

As mentioned in Section 3.2, larger mini-batches (increasing m) stabilize parameter updates, while larger stepsizes (decreasing α) provide swifter

DOC accuracy

$\alpha\backslash m$	1	3	10	30	100	300	1K	3K	10K
0.5	5.4	5.4	5.5	5.6	6.0	25.7	**48.8**	**49.9**	**44.6**
0.6	5.4	5.4	5.6	5.6	22.3	36.1	48.7	49.3	44.2
0.7	5.5	5.5	5.6	11.1	39.9	43.3	48.1	49.0	43.5
0.8	5.6	5.6	6.0	21.7	47.3	45.0	47.8	49.5	42.8
0.9	5.8	6.0	13.4	32.4	**48.7**	48.4	46.4	49.4	42.4
1.0	**6.2**	**11.8**	**19.6**	**35.2**	47.6	**49.5**	47.5	49.3	41.7



DOC log-likelihood

$\alpha\backslash m$	1	3	10	30	100	300	1K	3K	10K
0.5	-8.875	-8.71	-8.61	-8.555	-8.505	-8.172	-7.92	-7.906	**-7.916**
0.6	-8.604	-8.575	-8.54	-8.524	-8.235	-8.041	-7.898	-7.901	**-7.916**
0.7	-8.541	-8.533	-8.531	-8.354	-8.023	-7.943	-7.886	-7.896	-7.918
0.8	-8.519	-8.506	-8.493	-8.228	-7.933	-7.896	**-7.883**	**-7.89**	-7.922
0.9	-8.505	-8.486	-8.283	-8.106	**-7.91**	**-7.889**	-7.889	-7.891	-7.927
1.0	**-8.471**	**-8.319**	**-8.204**	**-8.052**	-7.919	**-7.889**	-7.892	-7.896	-7.937

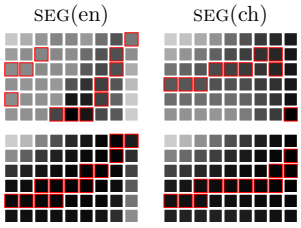


Figure 3: Effect of optimization parameters (stepsize reduction power α and mini-batch size m) on accuracy and likelihood. Numerical results are shown for document classification. In the interest of space, the results for each task are compressed into two gray scale images, one for accuracy (top) and one for log-likelihood (bottom), where darker shades represent larger values. Bold (red) numbers denote the best α for a given m.

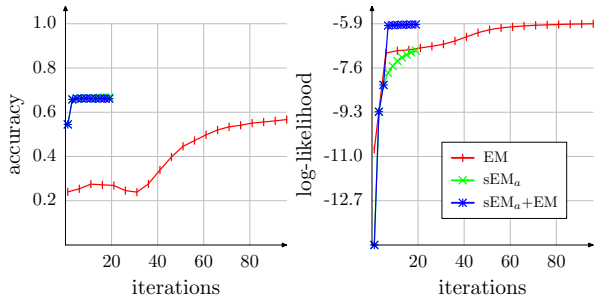


Figure 2: sEM$_a$ quickly obtains higher accuracy than batch EM but suffers from a slower increase in likelihood. The hybrid sEM$_a$+EM (5 iterations of EM$_a$ followed by batch EM) increases both accuracy and likelihood sharply.

progress. Remember that since we are dealing with a nonconvex objective, the choice of stepsize not only influences how fast we converge, but also the quality of the solution that we converge to.

Figure 3 shows the interaction between α and m in terms of likelihood and accuracy. In general, the best (α, m) depends on the task and dataset. For example, for document classification, larger m is critical for good performance; for POS tagging, it is better to use smaller values of α and m.

Fortunately, there is a range of permissible settings (corresponding to the dark regions in Figure 3) that lead to reasonable performance. Furthermore, the settings that perform well on likelihood generally correspond to ones that perform well on accuracy, which justifies using sEM$_\ell$.

A final observation is that as we use larger mini-batches (larger m), decreasing the stepsize more gradually (smaller α) leads to better performance. Intuitively, updates become more reliable with larger m, so we can afford to trust them more and incorporate them more aggressively.

Stepwise versus incremental EM In Section 3.2, we mentioned that incremental EM can be made equivalent to stepwise EM with $\alpha = 1$ and $m = 1$ (sEM$_i$). Figure 1 provides the empirical support: iEM and sEM$_i$ have very similar training curves. Therefore, keeping around the old sufficient statistics does not provide any advantage and still requires a substantial storage cost. As mentioned before, setting (α, m) properly is crucial. While we could simulate mini-batches with iEM by updating multiple coordinates simultaneously, iEM is not capable of exploiting the behavior of $\alpha < 1$.

4.4 Varying the random seed

All our results thus far represent single runs with a fixed random seed. We now investigate the impact of randomness on our results. Recall that we use randomness for two purposes: (1) initializing the parameters (affects both batch EM and online EM),

	accuracy		log-likelihood	
	EM	sEM$_\ell$	EM	sEM$_\ell$
POS	56.2 $\pm$1.36	58.8 $\pm$0.73, 1.41	-6.01	-6.09
DOC	41.2 $\pm$1.97	51.4 $\pm$0.97, 2.82	-7.93	-7.88
SEG(en)	80.5 $\pm$0.0	81.0 $\pm$0.0, 0.42	-4.1	-4.1
SEG(ch)	78.2 $\pm$0.0	77.2 $\pm$0.0, 0.04	-7.26	-7.27
ALIGN	79.0 $\pm$0.14	78.8 $\pm$0.14, 0.25	-5.04	-5.11

Table 2: Mean and standard deviation over different random seeds. For EM and sEM, the first number after $\pm$ is the standard deviation due to different initializations of the parameters. For sEM, the second number is the standard deviation due to different permutations of the examples. Standard deviation for log-likelihoods are all < 0.01 and therefore left out due to lack of space.

and (2) permuting the examples at the beginning of each iteration (affects only online EM).

To separate these two purposes, we used two different seeds, $S_i \in \{1,2,3,4,5\}$ and $S_p \in \{1,2,3,4,5\}$ for initializing and permuting, respectively. Let X be a random variable denoting either log-likelihood or accuracy. We define the variance due to initialization as $\text{var}(\mathbb{E}(X \mid S_i))$ ($\mathbb{E}$ averages over S_p for each fixed S_i) and the variance due to permutation as $\mathbb{E}(\text{var}(X \mid S_i))$ ($\mathbb{E}$ averages over S_i). These two variances provide an additive decomposition of the total variance: $\text{var}(X) = \text{var}(\mathbb{E}(X \mid S_i)) + \mathbb{E}(\text{var}(X \mid S_i))$.

Table 2 summarizes the results across the 5 trials for EM and 25 for sEM$_\ell$. Since we used a very small amount of noise to initialize the parameters, the variance due to initialization is systematically smaller than the variance due to permutation. sEM$_\ell$ is less sensitive to initialization than EM, but additional variance is created by randomly permuting the examples. Overall, the accuracy of sEM$_\ell$ is more variable than that of EM, but not by a large amount.

5 Discussion and related work

As datasets increase in size, the demand for online algorithms has grown in recent years. One sees this clear trend in the supervised NLP literature—examples include the Perceptron algorithm for tagging (Collins, 2002), MIRA for dependency parsing (McDonald et al., 2005), exponentiated gradient algorithms (Collins et al., 2008), stochastic gradient for constituency parsing (Finkel et al., 2008), just to name a few. Empirically, online methods are often faster by an order of magnitude (Collins et al., 2008), and it has been argued on theoretical grounds that the fast, approximate nature of online methods is a good fit given that we are interested in test performance, not the training objective (Bottou and Bousquet, 2008; Shalev-Shwartz and Srebro, 2008).

However, in the unsupervised NLP literature, online methods are rarely seen,[5] and when they are, incremental EM is the dominant variant (Gildea and Hofmann, 1999; Kuo et al., 2008). Indeed, as we have shown, applying online EM does require some care, and some variants (including incremental EM) can fail catastrophically in face of local optima. Stepwise EM provides finer control via its optimization parameters and has proven quite successful.

One family of methods that resembles incremental EM includes collapsed samplers for Bayesian models—for example, Goldwater et al. (2006) and Goldwater and Griffiths (2007). These samplers keep track of a sample of the latent variables for each example, akin to the sufficient statistics that we store in incremental EM. In contrast, stepwise EM does not require this storage and operates more in the spirit of a truly online algorithm.

Besides speed, online algorithms are of interest for two additional reasons. First, in some applications, we receive examples sequentially and would like to estimate a model in real-time, e.g., in the clustering of news articles. Second, since humans learn sequentially, studying online EM might suggest new connections to cognitive mechanisms.

6 Conclusion

We have explored online EM on four tasks and demonstrated how to use stepwise EM to overcome the dangers of stochasticity and reap the benefits of frequent updates and fast learning. We also discovered that stepwise EM can actually improve accuracy, a phenomenon worthy of further investigation. This paper makes some progress on elucidating the properties of online EM. With this increased understanding, online EM, like its batch cousin, could become a mainstay for unsupervised learning.

[5] Other types of learning methods have been employed successfully, for example, Venkataraman (2001) and Seginer (2007).

References

L. Bottou and O. Bousquet. 2008. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NIPS)*.

O. Cappé and E. Moulines. 2009. Online expectation-maximization algorithm for latent data models. *Journal of the Royal Statistics Society: Series B (Statistical Methodology)*, 71.

M. Collins, A. Globerson, T. Koo, X. Carreras, and P. Bartlett. 2008. Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *Journal of Machine Learning Research*, 9.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with Perceptron algorithms. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. R. Finkel, A. Kleeman, and C. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Human Language Technology and Association for Computational Linguistics (HLT/ACL)*.

D. Gildea and T. Hofmann. 1999. Topic-based language models using EM. In *Eurospeech*.

S. Goldwater and T. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Association for Computational Linguistics (ACL)*.

S. Goldwater, T. Griffiths, and M. Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL)*.

M. Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*.

M. Johnson. 2008. Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure. In *Human Language Technology and Association for Computational Linguistics (HLT/ACL)*, pages 398–406.

J. Kuo, H. Li, and C. Lin. 2008. Mining transliterations from web query results: An incremental approach. In *Sixth SIGHAN Workshop on Chinese Language Processing*.

P. Liang and D. Klein. 2008. Analyzing the errors of unsupervised learning. In *Human Language Technology and Association for Computational Linguistics (HLT/ACL)*.

P. Liang, D. Klein, and M. I. Jordan. 2008. Agreement-based learning. In *Advances in Neural Information Processing Systems (NIPS)*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Association for Computational Linguistics (ACL)*.

R. Neal and G. Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*.

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

M. Sato and S. Ishii. 2000. On-line EM algorithm for the normalized Gaussian network. *Neural Computation*, 12:407–432.

Y. Seginer. 2007. Fast unsupervised incremental parsing. In *Association for Computational Linguistics (ACL)*.

S. Shalev-Shwartz and N. Srebro. 2008. SVM optimization: Inverse dependence on training set size. In *International Conference on Machine Learning (ICML)*.

A. Venkataraman. 2001. A statistical model for word discovery in transcribed speech. *Computational Linguistics*, 27:351–372.

Unsupervised Approaches for Automatic Keyword Extraction Using Meeting Transcripts

Feifan Liu, Deana Pennell, Fei Liu and Yang Liu
Computer Science Department
The University of Texas at Dallas
Richardson, TX 75080, USA
{ffliu,deana,feiliu,yangl}@hlt.utdallas.edu

Abstract

This paper explores several unsupervised approaches to automatic keyword extraction using meeting transcripts. In the TFIDF (term frequency, inverse document frequency) weighting framework, we incorporated part-of-speech (POS) information, word clustering, and sentence salience score. We also evaluated a graph-based approach that measures the importance of a word based on its connection with other sentences or words. The system performance is evaluated in different ways, including comparison to human annotated keywords using F-measure and a weighted score relative to the oracle system performance, as well as a novel alternative human evaluation. Our results have shown that the simple unsupervised TFIDF approach performs reasonably well, and the additional information from POS and sentence score helps keyword extraction. However, the graph method is less effective for this domain. Experiments were also performed using speech recognition output and we observed degradation and different patterns compared to human transcripts.

1 Introduction

Keywords in a document provide important information about the content of the document. They can help users search through information more efficiently or decide whether to read a document. They can also be used for a variety of language processing tasks such as text categorization and information retrieval. However, most documents do not provide keywords. This is especially true for spoken documents. Current speech recognition system performance has improved significantly, but there is no rich structural information such as topics and keywords in the transcriptions. Therefore, there is a need to automatically generate keywords for the large amount of written or spoken documents available now.

There have been many efforts toward keyword extraction for text domain. In contrast, there is less work on speech transcripts. In this paper we focus on one speech genre — the multiparty meeting domain. Meeting speech is significantly different from written text and most other speech data. For example, there are typically multiple participants in a meeting, the discussion is not well organized, and the speech is spontaneous and contains disfluencies and ill-formed sentences. It is thus questionable whether we can adopt approaches that have been shown before to perform well in written text for automatic keyword extraction in meeting transcripts. In this paper, we evaluate several different keyword extraction algorithms using the transcripts of the ICSI meeting corpus. Starting from the simple TFIDF baseline, we introduce knowledge sources based on POS filtering, word clustering, and sentence salience score. In addition, we also investigate a graph-based algorithm in order to leverage more global information and reinforcement from summary sentences. We used different performance measurements: comparing to human annotated keywords using individual F-measures and a weighted score relative to the oracle system performance, and conducting novel human evaluation. Experiments were conducted using both the human transcripts and the speech recognition (ASR) out-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 620–628,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

put. Overall the TFIDF based framework seems to work well for this domain, and the additional knowledge sources help improve system performance. The graph-based approach yielded worse results, especially for the ASR condition, suggesting further investigation for this task.

2 Related Work

TFIDF weighting has been widely used for keyword or key phrase extraction. The idea is to identify words that appear frequently in a document, but do not occur frequently in the entire document collection. Much work has shown that TFIDF is very effective in extracting keywords for scientific journals, e.g., (Frank et al., 1999; Hulth, 2003; Kerner et al., 2005). However, we may not have a big background collection that matches the test domain for a reliable IDF estimate. (Matsuo and Ishizuka, 2004) proposed a co-occurrence distribution based method using a clustering strategy for extracting keywords for a single document without relying on a large corpus, and reported promising results.

Web information has also been used as an additional knowledge source for keyword extraction. (Turney, 2002) selected a set of keywords first and then determined whether to add another keyword hypothesis based on its PMI (point-wise mutual information) score to the current selected keywords. The preselected keywords can be generated using basic extraction algorithms such as TFIDF. It is important to ensure the quality of the first selection for the subsequent addition of keywords. Other researchers also used PMI scores between each pair of candidate keywords to select the top k% of words that have the highest average PMI scores as the final keywords (Inkpen and Desilets, 2004).

Keyword extraction has also been treated as a classification task and solved using supervised machine learning approaches (Frank et al., 1999; Turney, 2000; Kerner et al., 2005; Turney, 2002; Turney, 2003). In these approaches, the learning algorithm needs to learn to classify candidate words in the documents into positive or negative examples using a set of features. Useful features for this approach include TFIDF and its variations, position of a phrase, POS information, and relative length of a phrase (Turney, 2000). Some of these features may not work well for meeting transcripts. For example, the position of a phrase (measured by the number of words before its first appearance divided by the document length) is very useful for news article text, since keywords often appear early in the document (e.g., in the first paragraph). However, for the less well structured meeting domain (lack of title and paragraph), these kinds of features may not be indicative. A supervised approach to keyword extraction was used in (Liu et al., 2008). Even though the data set in that study is not very big, it seems that a supervised learning approach can achieve reasonable performance for this task.

Another line of research for keyword extraction has adopted graph-based methods similar to Google's PageRank algorithm (Brin and Page, 1998). In particular, (Wan et al., 2007) attempted to use a reinforcement approach to do keyword extraction and summarization simultaneously, on the assumption that important sentences usually contain keywords and keywords are usually seen in important sentences. We also find that this assumption also holds using statistics obtained from the meeting corpus used in this study. Graph-based methods have not been used in a genre like the meeting domain; therefore, it remains to be seen whether these approaches can be applied to meetings.

Not many studies have been performed on speech transcripts for keyword extraction. The most relevant work to our study is (Plas et al., 2004), where the task is keyword extraction in the multiparty meeting corpus. They showed that leveraging semantic resources can yield significant performance improvement compared to the approach based on the relative frequency ratio (similar to IDF). There is also some work using keywords for other speech processing tasks, e.g., (Munteanu et al., 2007; Bulyko et al., 2007; Wu et al., 2007; Desilets et al., 2002; Rogina, 2002). (Wu et al., 2007) showed that keyword extraction combined with semantic verification can be used to improve speech retrieval performance on broadcast news data. In (Rogina, 2002), keywords were extracted from lecture slides, and then used as queries to retrieve relevant web documents, resulting in an improved language model and better speech recognition performance of lectures. There are many differences between written text and speech — meetings in particular. Thus our goal in this paper is to investi-

gate whether we can successfully apply some existing techniques, as well as propose new approaches to extract keywords for the meeting domain. The aim of this study is to set up some starting points for research in this area.

3 Data

We used the meetings from the ICSI meeting data (Janin et al., 2003), which are recordings of naturally occurring meetings. All the meetings have been transcribed and annotated with dialog acts (DA) (Shriberg et al., 2004), topics, and extractive summaries (Murray et al., 2005). The ASR output for this corpus is obtained from a state-of-the-art SRI conversational telephone speech system (Zhu et al., 2005), with a word error rate of about 38.2% on the entire corpus. We align the human transcripts and ASR output, then map the human annotated DA boundaries and topic boundaries to the ASR words, such that we have human annotation of these information for the ASR output.

We recruited three Computer Science undergraduate students to annotate keywords for each topic segment, using 27 selected ICSI meetings.[1] Up to five indicative key words or phrases were annotated for each topic. In total, we have 208 topics annotated with keywords. The average length of the topics (measured using the number of dialog acts) among all the meetings is 172.5, with a high standard deviation of 236.8. We used six meetings as our development set (the same six meetings as the test set in (Murray et al., 2005)) to optimize our keyword extraction methods, and the remaining 21 meetings for final testing in Section 5.

One example of the annotated keywords for a topic segment is:

- **Annotator I**: analysis, constraints, template matcher;
- **Annotator II**: syntactic analysis, parser, pattern matcher, finite-state transducers;
- **Annotator III**: lexicon, set processing, chunk parser.

Note that these meetings are research discussions, and that the annotators may not be very familiar with the topics discussed and often had trouble deciding the important sentences or keywords. In addition, limiting the number of keywords that an annotator can select for a topic also created some difficulty. Sometimes there are more possible keywords and the annotators felt it is hard to decide which five are the most topic indicative. Among the three annotators, we notice that in general the quality of annotator I is the poorest. This is based on the authors' judgment, and is also confirmed later by an independent human evaluation (in Section 6).

For a better understanding of the gold standard used in this study and the task itself, we thoroughly analyzed the human annotation consistency. We removed the topics labeled with "chitchat" by at least one annotator, and also the digit recording part in the ICSI data, and used the remaining 140 topic segments. We calculated the percentage of keywords agreed upon by different annotators for each topic, as well as the average for all the meetings. All of the consistency analysis is performed based on words. Figure 1 illustrates the annotation consistency over different meetings and topics. The average consistency rate across topics is 22.76% and 5.97% among any two and all three annotators respectively. This suggests that people do not have a high agreement on keywords for a given document. We also notice that the two person agreement is up to 40% for several meetings and 80% for several individual topics, and the agreement among all three annotators reaches 20% and 40% for some meetings or topics. This implies that the consistency depends on topics (e.g., the difficulty or ambiguity of a topic itself, the annotators' knowledge of that topic). Further studies are needed for the possible factors affecting human agreement. We are currently creating more annotations for this data set for better agreement measure and also high quality annotation.

4 Methods

Our task is to extract keywords for each of the topic segments in each meeting transcript. Therefore, by "document", we mean a topic segment in the remainder of this paper. Note that our task is different from keyword spotting, where a keyword is provided and the task is to spot it in the audio (along with its transcript).

The core part of keyword extraction is for the sys-

[1] We selected these 27 meetings because they have been used in previous work for topic segmentation and summarization (Galley et al., 2003; Murray et al., 2005).

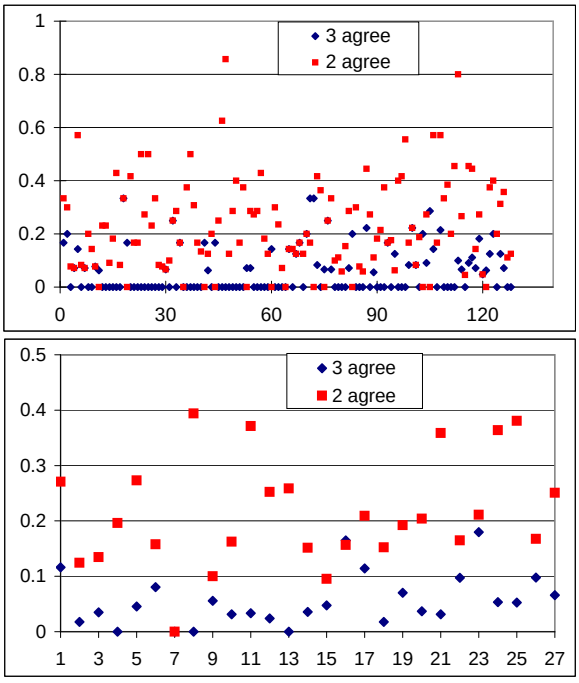


Figure 1: Human annotation consistency across different topics (upper graph) and meetings (lower graph). Y-axis is the percent of the keywords agreed upon by two or three annotators.

tem to assign an importance score to a word, and then pick the top ranked words as keywords. We compare different methods for weight calculation in this study, broadly divided into the following two categories: the TFIDF framework and the graph-based model. Both are unsupervised learning methods.[2] In all of the following approaches, when selecting the final keywords, we filter out any words appearing on the stopword list. These stopwords are generated based on the IDF values of the words using all the meeting data by treating each topic segment as a document. The top 250 words from this list (with the lowest IDF values) were used as stopwords. We generated two different stopword lists for human transcripts and ASR output respectively. In addition, in this paper we focus on performing keyword extraction at the single word level, therefore no key phrases are generated.

[2]Note that by unsupervised methods, we mean that no data annotated with keywords is needed. These methods do require the use of some data to generate information such as IDF, or possibly a development set to optimize some parameters or heuristic rules.

4.1 TFIDF Framework

(A) **Basic TFIDF weighting**

The term frequency (TF) for a word w_i in a document is the number of times the word occurs in the document. The IDF value is:

$$IDF_i = log(N/N_i)$$

where N_i denotes the number of the documents containing word w_i, and N is the total number of the documents in the collection. We also performed L_2 normalization for the IDF values when combining them with other scores.

(B) **Part of Speech (POS) filtering**

In addition to using a stopword list to remove words from consideration, we also leverage POS information to filter unlikely keywords. Our hypothesis is that verb, noun and adjective words are more likely to be keywords, so we restrict our selection to words with these POS tags only. We used the TnT POS tagger (Brants, 2000) trained from the Switchboard data to tag the meeting transcripts.

(C) **Integrating word clustering**

One weakness of the baseline TFIDF is that it counts the frequency for a particular word, without considering any words that are similar to it in terms of semantic meaning. In addition, when the document is short, the TF may not be a reliable indicator of the importance of the word. Our idea is therefore to account for the frequency of other similar words when calculating the TF of a word in the document. For this, we group all the words into clusters in an unsupervised fashion. If the total term frequency of all the words in one cluster is high, it is likely that this cluster contributes more to the current topic from a thematic point of view. Thus we want to assign higher weights to the words in this cluster.

We used the SRILM toolkit (Stolcke, 2002) for automatic word clustering over the entire document collection. It minimizes the perplexity of the induced class-based n-gram language model compared to the original word-based model. Using the clusters, we then adjust the TF weighting by integrating with the cluster term frequency (CTF):

$$TF\_CTF(w_i) = TF(w_i) * \alpha^{(\sum_{w_l \in C_i, w_l \neq w_i} freq(w_l))}$$

where the last summation component means the total term frequency of all the other words in this document that belong to the same cluster C_i as the current

word w_i. We set parameter α to be slightly larger than 1. We did not include stopwords when adding the term frequencies for the words in a cluster.

(D) **Combining with sentence salience score**

Intuitively, the words in an important sentence should be assigned a high weight for keyword extraction. In order to leverage the sentence information, we adjust a word's weight by the salience scores of the sentences containing that word. The sentence score is calculated based on its cosine similarity to the entire meeting. This score is often used in extractive summarization to select summary sentences (Radev et al., 2001). The cosine similarity between two vectors, D_1 and D_2, is defined as:

$$sim(D_1, D_2) = \frac{\sum_i t_{1i} t_{2i}}{\sqrt{\sum_i t_{1i}^2} \times \sqrt{\sum_i t_{2i}^2}}$$

where t_i is the term weight for a word w_i, for which we use the TFIDF value.

4.2 Graph-based Methods

For the graph-based approach, we adopt the iterative reinforcement approach from (Wan et al., 2007) in the hope of leveraging sentence information for keyword extraction. This algorithm is based on the assumption that important sentences/words are connected to other important sentences/words.

Four graphs are created: one graph in which sentences are connected to other sentences (S-S graph), one in which words are connected to other words (W-W graph), and two graphs connecting words to sentences with uni-directional edges (W-S and S-W graphs). Stopwords are removed before the creation of the graphs so they will be ineligible to be keywords.

The final weight for a word node depends on its connection to other words (W-W graph) and other sentences (W-S graph); similarly, the weight for a sentence node is dependent on its connection to other sentences (S-S graph) and other words (S-W graph). That is,

$$u = \alpha U^T u + \beta \hat{W}^T v$$
$$v = \alpha V^T v + \beta W^T u$$

where u and v are the weight vectors for sentence and word nodes respectively, $U, V, W, \hat{W}$ represent the S-S, W-W, S-W, and W-S connections. α and β specify the contributions from the homogeneous and the heterogeneous nodes. The initial weight is a uniform one for the word and sentence vector. Then the iterative reinforcement algorithm is used until the node weight values converge (the difference between scores at two iterations is below 0.0001 for all nodes) or 5,000 iterations are reached.

We have explored various ways to assign weights to the edges in the graphs. Based on the results on the development set, we use the following setup in this paper:

- **W-W Graph:** We used a diagonal matrix for the graph connection, i.e., there is no connection among words. The self-loop values are the TFIDF values of the words. This is also equivalent to using an identity matrix for the word-word connection and TFIDF as the initial weight for each vertex in the graph. We investigated other strategies to assign a weight for the edge between two word nodes; however, so far the best result we obtained is using this diagonal matrix.

- **S-W and W-S Graphs:** The weight for an edge between a sentence and a word is the TF of the word in the sentence multiplied by the word's IDF value. These weights are initially added only to the S-W graph, as in (Wan et al., 2007); then that graph is normalized and transposed to create the W-S graph.

- **S-S Graph:** The sentence node uses a vector space model and is composed of the weights of those words connected to this sentence in the S-W graph. We then use cosine similarity between two sentence vectors.

Similar to the above TFIDF framework, we also use POS filtering for the graph-based approach. After the weights for all the words are determined, we select the top ranked words with the POS restriction.

5 Experimental Results: Automatic Evaluation

Using the approaches described above, we computed weights for the words and then picked the top five words as the keywords for a topic. We chose five keywords since this is the number of keywords that

human annotators used as a guideline, and it also yielded good performance in the development set. To evaluate system performance, in this section we use human annotated keywords as references, and compare the system output to them. The first metric we use is F-measure, which has been widely used for this task and other detection tasks. We compare the system output with respect to each human annotation, and calculate the maximum and the average F-scores. Note that our keyword evaluation is word-based. When human annotators choose key phrases (containing more than one word), we split them into words and measure the matching words. Therefore, when the system only generates five keywords, the upper bound of the recall rate may not be 100%. In (Liu et al., 2008), a lenient metric is used which accounts for some inflection of words. Since that is highly correlated with the results using exact word match, we report results based on strict matching in the following experiments.

The second metric we use is similar to Pyramid (Nenkova and Passonneau, 2004), which has been used for summarization evaluation. Instead of comparing the system output with each individual human annotation, the method creates a "pyramid" using all the human annotated keywords, and then compares system output to this pyramid. The pyramid consists of all the annotated keywords at different levels. Each keyword has a score based on how many annotators have selected this one. The higher the score, the higher up the keyword will be in the pyramid. Then we calculate an oracle score that a system can obtain when generating k keywords. This is done by selecting keywords in the decreasing order in terms of the pyramid levels until we obtain k keywords. Finally for the system hypothesized k keywords, we compute its score by adding the scores of the keywords that match those in the pyramid. The system's performance is measured using the relative performance of the system's pyramid scores divided by the oracle score.

Table 1 shows the results using human transcripts for different methods on the 21 test meetings (139 topic segments in total). For comparison, we also show results using the supervised approach as in (Liu et al., 2008), which is the average of the 21-fold cross validation. We only show the maximum F-measure with respect to individual annotations, since the average scores show similar trend. In addition, the weighted relative scores already accounts for the different annotation and human agreement.

Methods	F-measure	weighted relative score
TFIDF	0.267	0.368
+ POS	0.275	0.370
+ Clustering	0.277	0.367
+ Sent weight	0.290	0.404
Graph	0.258	0.364
Graph+POS	0.277	0.380
Supervised	0.312	0.401

Table 1: Keyword extraction results using human transcripts compared to human annotations.

We notice that for the TFIDF framework, adding POS information slightly helps the basic TFIDF method. In all the meetings, our statistics show that adding POS filtering removed 2.3% of human annotated keywords from the word candidates; therefore, this does not have a significant negative impact on the upper bound recall rate, but helps eliminate unlikely keyword candidates. Using word clustering does not yield a performance gain, most likely because of the clustering technique we used — it does clustering simply based on word co-occurrence and does not capture semantic similarity properly.

Combining the term weight with the sentence salience score improves performance, supporting the hypothesis that summary sentences and keywords can reinforce each other. In fact we performed an analysis of keywords and summaries using the following two statistics:

$$(1) \qquad k = \frac{P_{summary}(w_i)}{P_{topic}(w_i)}$$

where $P_{summary}(w_i)$ and $P_{topic}(w_i)$ represent the the normalized frequency of a keyword w_i in the summary and the entire topic respectively; and

$$(2) \qquad s = \frac{PS_{summary}}{PS_{topic}}$$

where $PS_{summary}$ represents the percentage of the sentences containing at least one keyword among all the sentences in the summary, and similarly PS_{topic} is measured using the entire topic segment. We found that the average k and s are around 3.42 and 6.33 respectively. This means that keywords are

more likely to occur in the summary compared to the rest of the topic, and the chance for a summary sentence to contain at least one keyword is much higher than for the other sentences in the topic.

For the graph-based methods, we notice that adding POS filtering also improves performance, similar to the TFIDF framework. However, the graph method does not perform as well as the TFIDF approach. Comparing with using TFIDF alone, the graph method (without using POS) yielded worse results. In addition to using the TFIDF for the word nodes, information from the sentences is used in the graph method since a word is linked to sentences containing this word. The global information in the S-S graph (connecting a sentence to other sentences in the document) is propagated to the word nodes. Unlike the study in (Wan et al., 2007), this information does not yield any gain. We did find that the graph approach performed better in the development set, but it seems that it does not generalize to this test set.

Compared to the supervised results, the TFIDF approach is worse in terms of the individual maximum F-measure, but achieves similar performance when using the weighted relative score. However, the unsupervised TFIDF approach is much simpler and does not require any annotated data for training. Therefore it may be easily applied to a new domain. Again note that these results used word-based selection. (Liu et al., 2008) investigated adding bigram key phrases, which we expect to be independent of these unigram-based approaches and adding bigram phrases will yield further performance gain for the unsupervised approach. Finally, we analyzed if the system's keyword extraction performance is correlated with human annotation disagreement using the unsupervised approach (TFIDF+POS+Sent_weight). The correlation (Spearman's ρ value) between the system's F-measure and the three-annotator consistency on the 27 meetings is 0.5049 (p=0.0072). This indicates that for the meetings with a high disagreement among human annotators, it is also challenging for the automatic systems.

Table 2 shows the results using ASR output for various approaches. The performance measure is the same as used in Table 1. We find that in general, there is a performance degradation compared to using human transcripts, which is as expected. We found that only 59.74% of the human annotated keywords appear in ASR output, that is, the upper bound of recall is very low. The TFIDF approach still outperforms the graph method. Unlike on human transcripts, the addition of information sources in the TFIDF approach did not yield significant performance gain. A big difference from the human transcript condition is the use of sentence weighting — adding it degrades performance in ASR, in contrast to the improvement in human transcripts. This is possibly because the weighting of the sentences is poor when there are many recognition errors from content words. In addition, compared to the supervised results, the TFIDF method has similar maximum F-measure, but is slightly worse using the weighted score. Further research is needed for the ASR condition to investigate better modeling approaches.

Methods	F-measure	weighted relative score
TFIDF	0.191	0.257
+ POS	0.196	0.259
+ Clustering	0.196	0.259
+ Sent weigh	0.178	0.241
Graph	0.173	0.223
Graph+POS	0.183	0.233
Supervised	0.197	0.269

Table 2: Keyword extraction results using ASR output.

6 Experimental Results: Human Evaluation

Given the disagreement among human annotators, one question we need to answer is whether F-measure or even the weighted relative scores compared with human annotations are appropriate metrics to evaluate system-generated keywords. For example, precision measures among the system-generated keywords how many are correct. However, this does not measure if the unmatched system-generated keywords are bad or acceptable. We therefore performed a small scale human evaluation. We selected four topic segments from four different meetings, and gave output from different systems to five human subjects. The subjects ranged in age from 22 to 63, and all but one had only basic knowledge of computers. We first asked the eval-

uators to read the entire topic transcript, and then presented them with the system-generated keywords (randomly ordered by different systems). For comparison, the keywords annotated by our three human annotators were also included without revealing which sets of keywords were generated by a human and which by a computer. Because there was such disagreement between annotators regarding what made **good** keywords, we instead asked our evaluators to mark any words that were **definitely not** keywords. Systems that produced more of these rejected words (such as "basically" or "mmm-hm") are assumed to be worse than those containing fewer rejected words. We then measured the percentage of rejected keywords for each system/annotator. The results are shown in Table 3. Not surprisingly, the human annotations rank at the top. Overall, we find human evaluation results to be consistent with the automatic evaluation metrics in terms of the ranking of different systems.

Systems	Rejection rate
Annotator 2	8%
Annotator 3	19%
Annotator 1	25%
TFIDF + POS	28%
TFIDF	30%

Table 3: Human evaluation results: percentage of the rejected keywords by human evaluators for different systems/annotators.

Note this rejection rate is highly related to the recall/precision measure in the sense that it measures how many keywords are acceptable (or rejected) among the system generated ones. However, instead of comparing to a fixed set of human annotated keywords (e.g., five) and using that as a gold standard to compute recall/precision, in this evaluation, the human evaluator may have a larger set of acceptable keywords in their mind. We also measured the human evaluator agreement regarding the accepted or bad keywords. We found that the agreement on a bad keyword among five, four, and three human evaluator is 10.1%, 14.8%, and 10.1% respectively. This suggests that humans are more likely to agree on a bad keyword selection compared to agreement on the selected keywords, as discussed in Section 3 (even though the data sets in these two analysis are not the same). Another observation from the human evaluation is that sometimes a person rejects a keyword from one system output, but accepts that on the list from another system. We are not sure yet whether this is the inconsistency from human evaluators or whether the judgment is based on a word's occurrence with other provided keywords and thus some kind of semantic coherence. Further investigation on human evaluation is still needed.

7 Conclusions and Future Work

In this paper, we evaluated unsupervised keyword extraction performance for the meeting domain, a genre that is significantly different from most previous work. We compared several different approaches using the transcripts of the ICSI meeting corpus. Our results on the human transcripts show that the simple TFIDF based method is very competitive. Adding additional knowledge such as POS and sentence salience score helps improve performance. The graph-based approach performs less well in this task, possibly because of the lack of structure in this domain. We use different performance measurements, including F-measure with respect to individual human annotations and a weighted metric relative to the oracle system performance. We also performed a new human evaluation for this task and our results show consistency with the automatic measurement. In addition, experiments on the ASR output show performance degradation, but more importantly, different patterns in terms of the contributions of information sources compared to using human transcripts. Overall the unsupervised approaches are simple but effective; however, system performance compared to the human performance is still low, suggesting more work is needed for this domain.

For the future work, we plan to investigate different weighting algorithms for the graph-based approach. We also need a better way to decide the number of keywords to generate instead of using a fixed number. Furthermore, since there are multiple speakers in the meeting domain, we plan to incorporate speaker information in various approaches. More importantly, we will perform a more rigorous human evaluation, and also use extrinsic evaluation to see whether automatically generated keywords facilitate tasks such as information retrieval or meeting browsing.

Acknowledgments

This work is supported by NSF award IIS-0714132. Any opinions expressed in this work are those of the authors and do not necessarily reflect the views of NSF.

References

T. Brants. 2000. TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference*.

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30.

I. Bulyko, M. Ostendorf, M. Siu, T. Ng, A. Stolcke, and O. Cetin. 2007. Web resources for language modeling in conversational speech recognition. *ACM Transactions on Speech and Language Processing*, 5:1–25.

A. Desilets, B.D. Bruijn, and J. Martin. 2002. Extracting keyphrases from spoken audio documents. In *Information Retrieval Techniques for Speech Applications*, pages 339–342.

E. Frank, G.W. Paynter, I.H. Witten, C. Gutwin, and C.G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of IJCAI*, pages 688–673.

M. Galley, K. McKeown, E. Fosler-Lussier, and H. Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of ACL*.

A. Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of EMNLP*, pages 216–223.

D. Inkpen and A. Desilets. 2004. Extracting semantically-coherent keyphrases from speech. *Canadian Acoustics Association*, 32:130–131.

A. Janin, D. Baron, J. Edwards, D. Ellis, G . Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. 2003. The ICSI meeting corpus. In *Proceedings of ICASSP*.

Y.H. Kerner, Z. Gross, and A. Masa. 2005. Automatic extraction and learning of keyphrases from scientific articles. In *Computational Linguistics and Intelligent Text Processing*, pages 657–669.

F. Liu, F. Liu, and Y. Liu. 2008. Automatic keyword extraction for the meeting corpus using supervised approach and bigram expansion. In *Proceedings of IEEE SLT*.

Y. Matsuo and M. Ishizuka. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence*, 13(1):157–169.

C. Munteanu, G. Penn, and R. Baecker. 2007. Web-based language modeling for automatic lecture transcription. In *Proceedings of Interspeech*.

G. Murray, S. Renals, J. Carletta, and J. Moore. 2005. Evaluating automatic summaries of meeting recordings. In *Proceedings of ACL 2005 MTSE Workshop*, pages 33–40.

A. Nenkova and R. Passonneau. 2004. Evaluating content selection in summarization: the pyramid method. In *Proceedings of HLT/NAACL*.

L. Plas, V. Pallotta, M. Rajman, and H. Ghorbel. 2004. Automatic keyword extraction from spoken text. a comparison of two lexical resources: the EDR and WordNet. In *Proceedings of the LREC*.

D. Radev, S. Blair-Goldensohn, and Z. Zhang. 2001. Experiments in single and multi-document summarization using MEAD. In *Proceedings of The First Document Understanding Conference*.

I. Rogina. 2002. Lecture and presentation tracking in an intelligent meeting room. In *Proceedings of ICMI*.

E. Shriberg, R. Dhillon, S. Bhagat, J. Ang, and H. Carvey. 2004. The ICSI meeting recorder dialog act (MRDA) corpus. In *Proceedings of SIGDial Workshop*, pages 97–100.

A. Stolcke. 2002. SRILM – An extensible language modeling toolkit. In *Proceedings of ICSLP*, pages 901–904.

P.D. Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2:303–336.

P.D. Turney. 2002. Mining the web for lexical knowledge to improve keyphrase extraction: Learning from labeled and unlabeled data. In *National Research Council, Institute for Information Technology, Technical Report ERB-1096*.

P.D. Turney. 2003. Coherent keyphrase extraction via web mining. In *Proceedings of IJCAI*, pages 434–439.

X. Wan, J. Yang, and J. Xiao. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of ACL*, pages 552–559.

C.H. Wu, C.L. Huang, C.S. Hsu, and K.M. Lee. 2007. Speech retrieval using spoken keyword extraction and semantic verification. In *Proceedings of IEEE Region 10 Conference*, pages 1–4.

Q. Zhu, A. Stolcke, B. Chen, and N. Morgan. 2005. Using MLP features in SRI's conversational speech recognition system. In *Proceedings of Interspeech*.

A Finite-State Turn-Taking Model for Spoken Dialog Systems

Antoine Raux*
Honda Research Institute
800 California Street
Mountain View, CA 94041, USA
araux@hra.com

Maxine Eskenazi
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
max@cs.cmu.edu

Abstract

This paper introduces the Finite-State Turn-Taking Machine (FSTTM), a new model to control the turn-taking behavior of conversational agents. Based on a non-deterministic finite-state machine, the FSTTM uses a cost matrix and decision theoretic principles to select a turn-taking action at any time. We show how the model can be applied to the problem of end-of-turn detection. Evaluation results on a deployed spoken dialog system show that the FSTTM provides significantly higher responsiveness than previous approaches.

1 Introduction

Turn-taking, the process by which participants in a conversation alternate speech and silence, is an essential component of spoken interaction. In order to lead productive conversations, people need not only know *what* to say but also *when* to say it. Decades of research on Conversation Analysis and psycholinguistics (Duncan, 1972; Sacks et al., 1974; Oreström, 1983; Schegloff, 2000; Wesseling and van Son, 2005) have shown that human turn-taking behavior relies on a wide range of rules and signals at many different levels of language, from prosody to syntax, semantics, and discourse structure. In contrast, turn-taking in spoken dialog systems is often reduced to ad hoc rules only based on very low level features. This simplistic approach leads to inefficient, unnatural, and possibly confusing behavior (Porzel and Baudis, 2004; Ward et al., 2005).

Recently, more complex models of turn-taking have been proposed (Cassell et al., 2001; Thorisson, 2002; Kronild, 2006). Yet, these models still rely extensively on hand-coded expert knowledge and do not lend themselves to data-driven optimization. Furthermore, to our knowledge, no such model has been deployed in a widely used system outside of the laboratory. In this paper, we propose a flexible, practical model of turn-taking behavior that builds upon previous work on finite-state models of the conversational floor. Because of its simplicity and generality, this model can be applied to many turn-taking phenomena. At the same time, being grounded in decision theory, it lends itself well to data-driven optimization. We illustrate our approach by applying the model to a specific turn-taking task: end-of-turn detection.

2 Conversational Floor as a Finite-State Machine

2.1 6-state finite state models of turn-taking

In the 1960's and early 1970's, several researchers proposed models to explain the rhythmic turn-taking patterns in human conversation. In particular, Jaffe and Feldstein (1970) studied the mean duration of pauses, switching pauses (when a different speaker takes the floor), simultaneous speech, and (single-speaker) vocalizations in recorded dyadic conversations. Based on their observation that these durations follow exponential distributions, they proposed first-order Markov models to capture the alternation of speech and silence in dialog. Their initial model had four states: only participant A is speak-

*This research was conducted when the first author was a student at the Language Technologies Institute.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 629–637,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

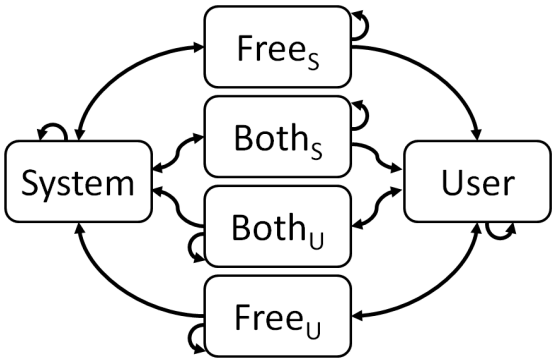


Figure 1: Our six-state model of turn-taking, inspired by Jaffe and Feldstein (1970) and Brady (1969). See section 3.1 for a description of the states.

ing; only participant B is speaking; both participants are speaking; and neither participant is speaking. However, such a model fails to distinguish switching pauses from A to B from switching pauses from B to A. Based on this observation, they extend their model to a six-state model which they found to better fit their data than the four-state model. Around the same time, Brady (1969) developed a very similar six-state model. He trained the parameters on a recorded conversation and compared the generated conversations to the original real one along several dimensions (pause and speech segment durations, overlaps, etc), finding that his model generally produced a good fit of the data.

2.2 Finite-State Models for Control

While Jaffe, Feldstein and Brady were primarily concerned with the analysis of human-human conversations, more recently, several researchers have proposed finite-state machines to control conversational agents. For instance, Cassell et al. (2001) model the conversational state of an embodied real estate agent as a 5-state machine. Two states indicate whether a user is present or not, whereas the other three indicate who holds the floor between the user and the agent, or whether the floor is open. Floor conflicts are not captured by this machine and are presumably resolved through simple rules (e.g. when the user speaks, the agent yields the floor).

Kronild (2006) proposes a much more complex model, based on Harel statecharts, which are an extension of finite-state machines for modeling and visualizing abstract control (Harel, 1987).

Thorisson's Ymir architecture (Thorisson, 2002) is an attempt to model the cognitive processes involved in conversation. It features dialog states, capturing, for example, who has the floor, and rules that govern the transition from one state to another based on "boolean conditions of perceptual features".

All these models are deterministic. At any point in time, the agent *knows* who owns the floor and uses fixed rules to take appropriate actions. These approaches assume 1) that the system can obtain perfectly reliable information on the state of the world, and 2) that the state itself is unambiguous.

3 The Finite-State Turn-Taking Machine

3.1 Extending the 6-state model for control

Our model, the Finite-State Turn-Taking Machine (FSTTM), uses the same six states as Jaffe and Feldstein: $USER$ and $SYSTEM$ represent states where one and only one of the participants claims the floor, $FREE_S$ and $FREE_U$ states where no participant claims the floor (following, resp., a $SYSTEM$ and $USER$ state), and $BOTH_S$ and $BOTH_U$ states where both participants claim the floor (following, resp. a $SYSTEM$ and $USER$ state). However, we apply this model to the control of a conversational agent, with a goal similar to that of Cassel, Thorisson, and Kronild. One important distinction is that we define the states in terms of the participants' *intentions* and *obligations* (in the sense of Traum and Allen (1994)) rather than the surface level observation of speech vs silence. For example, the state is $USER$ when the user has the obligation to speak (to respond to a system question) or the intention to speak, while at the same time, the system does not hold the floor. This does not necessarily mean that the user is speaking, for example at pauses during a user utterance.

As can be seen in Figure 1, not all transitions are valid. First, there is no direct transition between any of the intermediate states (the two $FREE$ states and two $BOTH$ states). The assumption is that to go from any of these state to another, the model will first go to either $SYSTEM$ or $USER$. This is an

approximation as there might be cases where, for example, both the system and user start speaking at the exact same time, going from a $FREE$ state to a $BOTH$ state. However these cases are rare enough that they can be approximated using a transition through either $SYSTEM$ or $USER$. Second, because intermediate states are conditioned on who had the floor previously, not all valid transitions are bidirectional. For example, there is no transition from $SYSTEM$ to $BOTH_U$. We associate pairs of user/system actions to each transition. The four possible actions are Grab the floor, Release the floor, Wait while not claiming the floor, and Keep the floor. For example, transition from $SYSTEM$ to $FREE_S$ corresponds to the user waiting silently and the system releasing the floor at the end of a prompt, noted (R, W) (we always note the system action first and user action second).

This representation allows us to formalize a wide variety of turn-taking phenomena in a unified framework. Specifically, there are 4 types of 2-step transitions from a single-floor-holder state ($SYSTEM$ or $USER$) to another (or the same) single-floor-holder state, which represent typical turn-taking phenomena:

Turn transitions with gap are the most common way the floor goes from one participant to the other. For example, at the end of a user utterance, once the user finishes speaking, the floor becomes free, after which the system starts responding, thus grabbing the floor. The resulting state sequence is:

$$SYSTEM \overset{(R,W)}{\rightarrow} FREE_S \overset{(W,G)}{\rightarrow} USER$$

Conversely, the transition with gap following a system prompt corresponds to:

$$USER \overset{(R,W)}{\rightarrow} FREE_S \overset{(W,G}{\rightarrow} USER$$

Turn transitions with overlap happen when a participant grabs the floor while it still belongs to the other. For example, when a user barges in on a system prompt, both participants hold the floor. Then, the system recognizes the barge-in attempt and relinquishes the floor, which becomes user's.

$$SYSTEM \overset{(K,G)}{\rightarrow} BOTH_S \overset{(R,K}{\rightarrow} USER$$

And conversely, when the system interrupts the user mid-utterance (which in dialog systems is more often the result of an intentional cut-in, rather than intentional interruption), the state sequence is:

$$USER \overset{(G,K)}{\rightarrow} BOTH_U \overset{(K,R)}{\rightarrow} SYSTEM$$

Failed interruptions happen when a participant barges in on the other and then withdraws before the original floor holder releases the floor. For example, when the system interrupts the user (often by mistake) but detects it and interrupts itself:

$$USER \overset{(G,K)}{\rightarrow} BOTH_U \overset{(R,K}{\rightarrow} USER$$

The converse is usually the result of the system failing to react fast enough to a user barge-in:

$$SYSTEM \overset{(K,G)}{\rightarrow} BOTH_S \overset{(K,R)}{\rightarrow} SYSTEM$$

Note that backchannels seem to fit in this category too. However, since backchannels, by definition, do not represent an attempt to grab the floor, they are not captured by the model as it is (for example, the floor should remain $SYSTEM$ when a user backchannels a system utterance).

Time outs start like transitions with gap but the intended next speaker (e.g. the user after a system prompt) does not take the floor and the original floor holder grabs it back. For instance, after a system prompt, if the floor remains free for a certain amount of time, the system attempts to re-establish the communication with the user, as follows:

$$SYSTEM \overset{(R,W)}{\rightarrow} FREE_S \overset{(G,W}{\rightarrow} SYSTEM$$

The opposite also happens when the system is to slow to respond to the user:

$$USER \overset{(W,R)}{\rightarrow} FREE_U \overset{(W,G}{\rightarrow} USER$$

While all the transitions above were described as deterministic, the actual state of the model is not fully observable. Specifically, while the system

knows whether its claiming the floor or not, it can only *believe* with some degree of uncertainty that the user does so. The system's knowledge of its own claim to the floor splits the state space into two disjoint subsets. When the system claims the floor, the state can be $SYSTEM$, $BOTH_S$, or $BOTH_U$). When the system does not claim the floor, the state can be $USER$, $FREE_U$, or $FREE_S$). In either case, the system needs to recognize the user's intention (i.e. whether the user claims to the floor or not) to maintain a probability distribution over the three states. Since the distinction between the two $BOTH$ states (resp. the two $FREE$ states) is based on past history that can be known with a high level of certainty, the uncertainty in state distribution is fully characterized by the probability that the user is claiming the floor, which will have to be estimated from observations, as we will see below.

3.2 Cost of Turn-Taking Actions

The problem we are facing is that of choosing the best system action given the system's belief about the current state of the model. That is achieved by applying the probabilistic decision theory principle of selecting the action with lowest expected cost. The actions available to the system are the four described above (G,R,K,W), although not all actions are available in all states. In fact, as can be seen in Table 1, there are always only two actions available in each state, depending on whether the system is claiming the floor or not.

Each action in each state has a particular cost. While there are many possible ways of defining these costs, we propose a simple cost structure that derives from the principles laid out in Sacks et al. (1974):

> Participants in a conversation attempt to minimize gaps and overlaps.

From this general principle, we derive three rules to drive the design of a cost matrix:

1. The cost of an action that resolves either a gap or an overlap is zero
2. The cost of an action that creates unwanted gap or overlap is equal to a constant parameter (potentially different for each action/state pair)
3. The cost of an action that maintains a gap or overlap is either a constant or an increasing function of the total time spent in that state

The resulting cost matrix is shown in Table 1, where

- $\boldsymbol{C_S}$ is the cost of interrupting a system prompt before its end when the user is not claiming the floor (false interruption)
- $\boldsymbol{C_O(\tau)}$ is the cost of remaining in an overlap that is already τ ms long
- $\boldsymbol{C_U}$ is the cost of grabbing the floor when the user is holding it (cut-in)
- $\boldsymbol{C_G(\tau)}$ is the cost of remaining in a gap that is already τ ms long

This cost structure makes a number of simplifying assumptions and there are many other possible cost matrices. For example, the cost of interrupting the user might vary depending on what has already been said in the utterance, so does the cost of interrupting a system prompt. A more principled approach to setting the costs would be to estimate from perceptual experiments or user studies what the impact of remaining in gap or overlap is compared to that of a cut-in or false interruption. However, as a first approximation, the proposed cost structure offers a simple way to take into account some of the constraints of interaction.

3.3 Decision Theoretic Action Selection

Given the state space and the cost matrix given above, the optimal decision at any point in time is the one that yields the lowest expected cost, where the expected cost of action A is:

$$C(A) = \sum_{S \in \Sigma} P(s = S|O) \cdot C(A, S)$$

where Σ is the set of states, O are the observable features of the world, and $C(A, S)$ is the cost of action A in state S, from the cost matrix in Table 1. In addition to the cost matrix' four constants, which we will consider as parameters of the model, it is thus necessary to estimate $P(s = S|O)$, which as seen above amounts to estimate the probability that the user is claiming the floor. Key to applying the FSTTM to a practical turn-taking problem is thus the construction of accurate estimates of the probabilities $P(s = S|O)$.

State \ Action	K	R	W	G
$SYSTEM$	0	C_S	-	-
$BOTH_S$	$C_O(\tau)$	0	-	-
$BOTH_U$	$C_O(\tau)$	0	-	-
$USER$	-	-	0	C_U
$FREE_U$	-	-	$C_G(\tau)$	0
$FREE_S$	-	-	$C_G(\tau)$	0

Table 1: Cost of system actions in each state (K: keep the floor, R: release the floor, W: wait without the floor, G: grab the floor, τ: time spent in current state, -: action unavailable).

4 Endpointing with the FSTTM

4.1 Problem formalization

In our FSTTM formalism, endpointing is the problem of selecting between the Wait and the Grab actions during a user utterance. We make the simplifying assumption that, once a user utterance has been detected, the only states with non-zero probability are $USER$ and $FREE_U$. While this does not capture cases where the system erroneously detects user speech (because there is, for example, background noise), it represents a working first approximation of the problem.

The main issue is to estimate the probability $P(s = FREE_U|O_t)$ (hereafter abbreviated as $P(F|O_t)$, $P(s = USER|O_t)$ being abbreviated as $P(U|O_t)$) where O_t represents all observable features at time t. Given that probability, the expected cost of grabbing the floor is:

$$\begin{aligned} C(G|O_t) &= P(U|O_t) \cdot C_U + P(F|O_t) \cdot 0 \\ &= (1 - P(F|O_t)) \cdot C_U \end{aligned}$$

Similarly, the expected cost of waiting is:

$$C(W|O_t) = P(F|O_t) \cdot C_G(\tau)$$

The system endpoints whenever the expected cost of grabbing the floor becomes higher than that of waiting.

We consider two separate cases for computing both $P(F|O_t)$ and $C_G(\tau)$: when a pause has been detected by the voice activity detector (VAD), and when no pause has been detected (yet). In the following sections, we provide details on the approximations and estimation methods for these two cases.

4.2 At pauses

If a pause has been detected by the VAD, we set the cost of waiting in the $FREE_U$ state to be proportional to the duration of the pause so far. If the user has released the floor, the duration of the current pause corresponds to the time spent in the $FREE_U$ state, i.e. τ in the cost matrix of Table 1. In this case, we set $C_G(\tau) = C_G^p \cdot \tau$ as a simple application of rule 3 from section 3.2.

We decompose the observations at time t, O_t, into observations available at the start of the pause (O), and observations made *during* the pause. With only audio information available, the only information available during the pause is its duration so far, i.e. τ. Specifically, we know that $d \geq \tau$, where d is the total duration of the pause (with $d = \infty$ at the end of a turn[1]). Consequently, $P(F|O_t)$ can be rewritten using Bayes rule as

$$\begin{aligned} P(F|O_t) &= \frac{P(d \geq \tau|O, F) \cdot P(F|O)}{P(d \geq \tau|O)} \\ &= \frac{P(F|O)}{P(d \geq \tau|O)} \end{aligned}$$

where $P(F|O)$ is the probability that the user released the floor without any knowledge of the duration of the pause, and $P(d \geq \tau|O)$ is the probability that the pause will last at least τ ms. We further decompose $P(d \geq \tau|O)$ into

$$P(d \geq \tau|O) = P(d \geq \tau, U|O) + P(d \geq \tau, F|O)$$

[1]Note that this is an approximation since the user could start speaking again after releasing the floor to reestablish the channel (e.g. by saying "Hello?"). However, in the vast majority of cases, the time after which the user resumes speaking is significantly longer than the time the system takes to endpoint.

$$
\begin{aligned}
&= P(d \geq \tau|O,U) \cdot P(U|O) + \\
&\quad P(d \geq \tau|O,F) \cdot P(F|O) \\
&= P(d \geq \tau|O,U) \cdot (1 - P(F|O)) \\
&\quad + P(F|O)
\end{aligned}
$$

Consequently, $P(F|O_t)$ is a function of $P(F|O)$ and $P(d \geq \tau|O,U)$. We estimate $P(F|O)$ by stepwise logistic regression on a training set of pauses labeled for finality (whether the pause is turn-final or turn-internal), using a wide range of features available from various components of the dialog system. Based on the well established observation that pause durations follow an exponential distribution (Jaffe and Feldstein, 1970; Lennes and Anttila, 2002; Raux et al., 2008), $P(d \geq \tau|O,U)$ is a function of mean pause duration, computed on the training set.

4.3 In speech

In some cases, it is not necessary to wait for the VAD to detect a pause to know with high confidence that the user has released the floor. For example, after a simple yes/no question, if the user says "YES", they are very likely to have released the floor, regardless of how long they remain silent afterwards. In order to exploit this fact and improve the responsiveness of the system in these highly predictable cases, we use a separate model to compute the expected costs of waiting and grabbing the floor before any pause is detected by the VAD (specifically, whenever the duration of the current pause is between 0 and 200 ms). In this case, we set the cost of waiting to a constant C_G^s. We train a logistic regression model to estimate $P(F|O_t)$ each time a new partial hypothesis is produced by the ASR during a user utterance. We use the same set of features as above.

5 Evaluation

5.1 Corpus and Features

We evaluated the effectiveness of the FSTTM on an actual deployed spoken dialog system. The system provides bus schedule information for a mid-size North American city. It is actually used by the general public and therefore constantly operates and collects data. In order to train the various probability estimation models and evaluate the approach in batch, we first collected a corpus of 586 dialogs between May 4, and May 14, 2008 (the "2008 corpus").

All of the features we used can be automatically extracted at runtime, and most of them were readily available in the system. They include dialog state information, turn-taking features, such as whether the current user utterance is a barge-in, and semantic information derived from the dialog state and partial recognition hypotheses provided by the speech recognizer. Dialog state is abstracted to three high-level states, which correspond to the type of system prompt directly preceding the user utterance: Open question ("What can I do for you?"); Closed question (e.g. "Where do you want to go?"); and Confirmation (e.g. "Going to the airport. Is this correct?").

To capture lexical cues correlated with the end of turns, we created a new feature called the boundary LM score. To compute it, we used previously collected data to train dialog-state-dependent statistical language models to estimate the probability that the hypothesis is complete. Boundary LM score is defined as the ratio of the log likelihood of the hypothesis being complete by that of the hypothesis being incomplete.

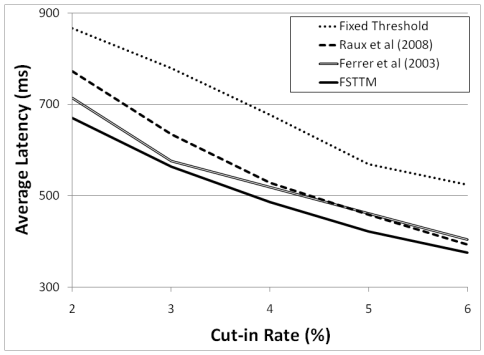
5.2 Estimating $P(F|O_t)$

We trained two logistic regression models using stepwise regression and 10-fold cross-validation for evaluation. The first model, whose performance is given in Table 2, estimates $P(F|O)$ at pauses. The model is unable to improve classification accuracy over the majority baseline for each state, however, the statistically significant improvement in average log likelihood indicates that the probability estimates are improved by using the features. The most informative feature in all three states was the boundary LM score introduced in section 5.1. Other selected features included the average number of words per user utterance so far and whether the current utterance is a barge-in (for the Open and Closed question states), as well as whether the partial hypothesis contained a confirmation marker such as "YES" or "SURE" (for the Confirmation state).

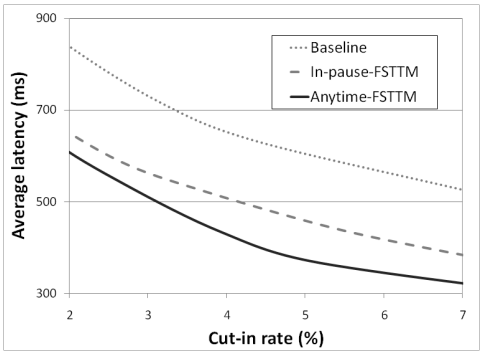
The second model performs the same regression, this time on all partial hypotheses received during speech segments. As seen in the "S" columns in Table 2, classification error was significantly reduced and the gain in average log likelihood were larger

	Open question		Closed question		Confirmation	
	P	S	P	S	P	S
Majority Baseline	38%	20%	25%	32%	12%	36%
Classification Error	35%	17%	26%	22%	12%	17%
Baseline log likelihood	-0.66	-0.50	-0.56	-0.63	-0.36	-0.65
Log likelihood	-0.61	-0.40	-0.50	-0.49	-0.30	-0.40

Table 2: Performance of state-specific logistic regression for estimating $P(F|O)$ at pauses (P) and in speech (S).



(a) In-pause evaluation on the 2007 corpus.



(a) Anytime evaluation on the 2008 corpus.

Figure 2: Batch evaluation of FSTTM endpointing.

than at pauses, particularly for the "Closed question" and "Confirmation" states. Again, boundary LM score was the most informative feature. The duration of the pause at the end of the partial hypothesis (between 0 and 200 ms) also proved well correlated with finality.

5.3 Batch Evaluation of the FSTTM

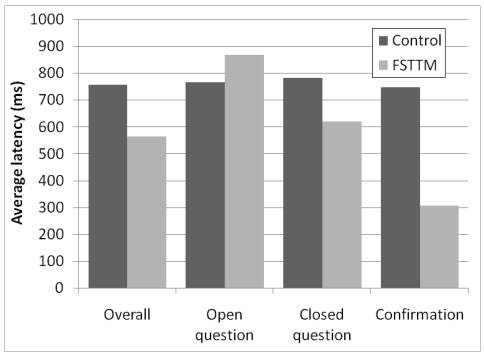
We performed two batch evaluations of the FSTTM. The first one aims at comparing in-pause-FSTTM with a fixed-threshold baseline as well as previous data-driven endpointing methods proposed in Ferrer et al. (2003) (reimplemented by us) and Raux et al. (2008). This evaluation was done on the corpus used in Raux et al. (2008) (the "2007 corpus"). As seen in Figure 2 (a), the FSTTM outperforms all other approaches (albeit only slightly compared to Ferrer et al.), improving over the fixed threshold baseline by up to 29.5%.

Second, we compared the anytime-FSTTM with in-pause-FSTTM and a fixed-threshold baseline (for reference) on the more recent 2008 corpus (since the 2007 corpus did not contain all necessary features for anytime-FSTTM). We set $C_G^p = 1$ and set C_G^s to either 0, leading to an endpointer that never endpoints during speech (in-pause-FSTTM), or 1000 (anytime-FSTTM). In both cases, we vary C_U to compute the latency / cut-in rate trade-off curve. The results are shown in Figure 2 (b). Anytime-FSTTM endpointing is consistently better than in-pause-FSTTM. For example, at a cut-in rate of 5%, anytime-FSTTM yields latencies that are on average 17% shorter than in-pause-FSTTM, and 40% shorter than the baseline. Additionally, we found that, in anytime-FSTTM, 30 to 40% of the turns are endpointed before the pause is detected by the VAD.

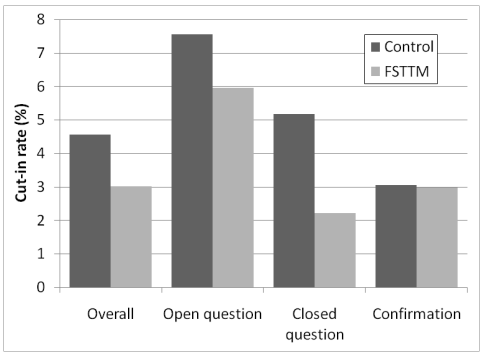
5.4 Live Evaluation

To confirm the results of the batch evaluation, we implemented our FSTTM model in the deployed system a let it run for ten days using either FSTTM or a fixed threshold for endpointing, resulting in a corpus of 171 FSTTM and 148 control dialogs. For FSTTM, we set $C_G^p = 1$, $C_G^s = 500$, and $C_U = 5000$. In the batch evaluation, these values correspond to a cut-in rate of 6.3% and an average latency of 320 ms. For the control condition, we set the fixed endpointing threshold to 555 ms, which also corresponded to about 6.3% cut-ins.

Figure 3 shows the average latency and cut-in rate



(a) Latency



(b) Cut-in rates

Figure 3: Live evaluation results. All confidence intervals for latency (not shown on the figure) fall within $+/-4ms$.

for both conditions. The FSTTM improves over the baseline on all metrics, reducing average latency by 193 ms ($p < 0.05$), cut-in rate by 1.5% (although this result is not statistically significant).

6 Discussion

Both batch and live evaluation results confirm the effectiveness of the FSTTM approach in improving system responsiveness. This approach significantly reduced endpointing latency over previous approaches. Boundary LM score got the highest weight in the regression, indicating that in a domain such as telephone-based information access, lexical cues are very informative for endpointing. The fact that boundary LMs can be computed without any human transcription effort (since they are trained on ASR output) makes them all the more appealing.

Essentially, the FSTTM provides a simple, unified model of turn-taking that lends itself to data-driven optimization. While we discussed specific cost structures and probability estimation techniques, the framework's flexibility opens it to other choices at many levels. By formalizing the overall turn-taking process in a probabilistic, decision-theoretic framework, the FSTTM extends and generalizes previous classification-based approaches to endpointing such as those proposed by Sato et al. (2002), Ferrer et al. (2003), Takeuchi et al. (2004), and our previous work (Raux et al., 2008).

Possible extensions of the approach include data-driven cost matrices to relax some of the assumptions introduced in section 3.2, as well as more complex state structures to handle, for example, multi-party conversations.

Finally, we plan to investigate more principled approaches, such as Partially Observable Markov Decision Processes or Dynamic Bayesian Networks, to model the different sources of uncertainty (detection errors and inherent ambiguity) and track the state distribution over time. Raux (2009) provides more details on all aspects of the approach and its possible extensions.

7 Conclusion

In this paper, motivated by existing finite-state models of turn-taking in dyadic conversations, we propose the Finite-State Turn-Taking Machine, an approach to turn-taking that relies on three core elements: a non-deterministic finite-state machine that captures the conversational floor; a cost matrix that models the impact of different system actions in different states; and a decision-theoretic action selection mechanism. We describe the application of the FSTTM to the key turn-taking phenomenon of end-of-turn detection. Evaluation both offline and by applying the FSTTM to a deployed spoken dialog system system showed that it performs significantly better than a fixed-threshold baseline.

Acknowledgments

This work is supported by the US National Science Foundation under grant number 0208835. 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. We would like to thank Alan Black for his many comments and advice.

References

P. T. Brady. 1969. A model for generating on-off speech patterns in two-way conversation. *The Bell System Technical Journal*, 48:2445–2472.

J. Cassell, T. Bickmore, L. Campbell, H. Vilhjalmsson, and H. Yan. 2001. More than just a pretty face: conversational protocols and the affordances of embodiment. *Knowledge-Based Systems*, 14:55–64.

S. Duncan. 1972. Some signals and rules for taking speaking turns in conversations. *Journal of Personality and Social Psychology*, 23(2):283–292.

L. Ferrer, E. Shriberg, and A. Stolcke. 2003. A prosody-based approach to end-of-utterance detection that does not require speech recognition. In *ICASSP*, Hong Kong.

D. Harel. 1987. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8:231–274.

J. Jaffe and S. Feldstein. 1970. *Rhythms of Dialogue*. Academic Press.

F. Kronild. 2006. Turn taking for artificial conversational agents. In *Cooperative Information Agents X*, Edinburgh, UK.

Mietta Lennes and Hanna Anttila. 2002. Prosodic features associated with the distribution of turns in finnish informal dialogues. In Petri Korhonen, editor, *The Phonetics Symposium 2002*, volume Report 67, pages 149–158. Laboratory of Acoustics and Audio Signal Processing, Helsinki University of Technology.

B. Oreström. 1983. *Turn-Taking in English Conversation*. CWK Gleerup, Lund.

R. Porzel and M. Baudis. 2004. The tao of chi: Towards effective human-computer interaction. In *HLT/NAACL 2004*, Boston, MA.

A. Raux, , and M. Eskenazi. 2008. Optimizing endpointing thresholds using dialogue features in a spoken dialogue system. In *Proc. SIGdial 2008*, Columbus, OH, USA.

A. Raux. 2009. *Flexible Turn-Taking for Spoken Dialog Systems*. Ph.D. thesis, Carnegie Mellon University.

H. Sacks, E. A. Schegloff, and G. Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735.

R. Sato, R. Higashinaka, M. Tamoto, M. Nakano, and K. Aikawa. 2002. Learning decision trees to determine turn-taking by spoken dialogue systems. In *ICSLP 2002*, Denver, CO.

E.A. Schegloff. 2000. Overlapping talk and the organization of turn-taking for conversation. *Language in Society*, 29:1–63.

M. Takeuchi, N. Kitaoka, and S. Nakagawa. 2004. Timing detection for realtime dialog systems using prosodic and linguistic information. In *Proc. Speech Prosody 04*, Nara, Japan.

K. R. Thorisson, 2002. *Multimodality in Language and Speech Systems*, chapter Natural Turn-Taking Needs No Manual: Computational Theory and Model, From Perception to Action, pages 173–207. Kluwer Academic Publishers.

D. R. Traum and J. F. Allen. 1994. Discourse obligations in dialogue. In *Proc. ACL-94*, pages 1–8.

N. Ward, A. Rivera, K. Ward, and D. Novick. 2005. Root causes of lost time and user stress in a simple dialog system. In *Interspeech 2005*, Lisbon, Portugal.

W. Wesseling and R.J.J.H. van Son. 2005. Timing of experimentally elicited minimal responses as quantitative evidence for the use of intonation in projecting TRPs. In *Interspeech 2005*, pages 3389–3392, Lisbon, Portugal.

Extracting Social Meaning: Identifying Interactional Style in Spoken Conversation

Dan Jurafsky
Linguistics Department
Stanford University
jurafsky@stanford.edu

Rajesh Ranganath
Computer Science Department
Stanford University
rajeshr@cs.stanford.edu

Dan McFarland
School of Education
Stanford University
dmcfarla@stanford.edu

Abstract

Automatically extracting social meaning and intention from spoken dialogue is an important task for dialogue systems and social computing. We describe a system for detecting elements of **interactional style**: whether a speaker is *awkward*, *friendly*, or *flirtatious*. We create and use a new spoken corpus of 991 4-minute speed-dates. Participants rated their interlocutors for these elements of style. Using rich dialogue, lexical, and prosodic features, we are able to detect flirtatious, awkward, and friendly styles in noisy natural conversational data with up to 75% accuracy, compared to a 50% baseline. We describe simple ways to extract relatively rich dialogue features, and analyze which features performed similarly for men and women and which were gender-specific.

1 Introduction

How can we extract social meaning from speech, deciding if a speaker is particularly engaged in the conversation, is uncomfortable or awkward, or is particularly friendly and flirtatious? Understanding these meanings and how they are signaled in language is an important sociolinguistic task in itself. Extracting them automatically from dialogue speech and text is crucial for developing socially aware computing systems for tasks such as detection of interactional problems or matching conversational style, and will play an important role in creating more natural dialogue agents (Pentland, 2005; Nass and Brave, 2005; Brave et al., 2005).

Cues for social meaning permeate speech at every level of linguistic structure. Acoustic cues such as low and high F0 or energy and spectral tilt are important in detecting emotions such as *annoyance*, *anger*, *sadness*, or *boredom* (Ang et al., 2002; Lee and Narayanan, 2002; Liscombe et al., 2003), speaker characteristics such as *charisma* (Rosenberg and Hirschberg, 2005), or *personality* features like extroversion (Mairesse et al., 2007; Mairesse and Walker, 2008). Lexical cues to social meaning abound. Speakers with links to depression or speakers who are under stress use more first person singular pronouns (Rude et al., 2004; Pennebaker and Lay, 2002; Cohn et al., 2004), positive emotion words are cues to agreeableness (Mairesse et al., 2007), and negative emotion words are useful cues to deceptive speech (Newman et al., 2003). The number of words in a sentence can be a useful feature for extroverted personality (Mairesse et al., 2007). Finally, dialog features such as the presence of disfluencies can inform listeners about speakers' problems in utterance planning or about confidence (Brennan and Williams, 1995; Brennan and Schober, 2001).

Our goal is to see whether cues of this sort are useful in detecting particular elements of conversational style and social intention; whether a speaker in a speed-dating conversation is judged by the interlocutor as *friendly*, *awkward*, or *flirtatious*.

2 The Corpus

Our experiments make use of a new corpus we have collected, the SpeedDate Corpus. The corpus is based on three speed-dating sessions run at an elite

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 638–646,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

private American university in 2005 and inspired by prior speed-dating research (Madan et al., 2005; Pentland, 2005). The graduate student participants volunteered to be in the study and were promised emails of persons with whom they reported mutual liking. Each date was conducted in an open setting where there was substantial background noise. All participants wore audio recorders on a shoulder sash, thus resulting in two audio recordings of the approximately 1100 4-minute dates. In addition to the audio, we collected pre-test surveys, event scorecards, and post-test surveys. This is the largest sample we know of where audio data and detailed survey information were combined in a natural experiment.

The rich survey information included date perceptions and follow-up interest, as well as general attitudes, preferences, and demographic information. Participants were also asked about the conversational style and intention of the interlocutor. Each speaker was asked to report how often their date's speech reflected different conversational styles (awkward, friendly, flirtatious, funny, assertive) on a scale of 1-10 (1=never, 10=constantly): "How often did the other person behave in the following ways on this 'date'?". We chose three of these five to focus on in this paper.

We acquired acoustic information by taking the acoustic wave file from each recorder and manually segmenting it into a sequence of wavefiles, each corresponding to one 4-minute date. Since both speakers wore microphones, most dates had two recordings, one from the male recorder and one from the female recorder. Because of mechanical, operator, and experimenter errors, some recordings were lost, and thus some dates had only one recording. Transcribers at a professional transcription service used the two recordings to create a transcript for each date, and time-stamped the start and end time of each speaker turn. Transcribers were instructed to mark various disfluencies as well as some nonverbal elements of the conversation such as laughter.

Because of noise, participants who accidentally turned off their mikes, and some segmentation and transcription errors, a number of dates were not possible to analyze. 19 dates were lost completely, and for an additional 130 we lost one of the two audio tracks and had to use the remaining track to extract features for both interlocutors. The current study focuses on the 991 remaining clean dates for which we had usable audio, transcripts, and survey information.

3 The Experiments

Our goal is to detect three of the style variables, in particular awkward, friendly, or flirtatious speakers, via a machine learning classifier. Recall that each speaker in a date (each conversation side) was labeled by his or her interlocutor with a rating from 1-10 for awkward, friendly, or flirtatious behavior. For the experiments, the 1-10 Likert scale ratings were first mean-centered within each respondent so that the average was 0. Then the top ten percent of the respondent-centered meaned Likert ratings were marked as positive for the trait, and the bottom ten percent were marked as negative for a trait. Thus each respondent labels the other speaker as either positive, negative, or NA for each of the three traits.

We run our binary classification experiments to predict this output variable.

For each speaker side of each 4-minute conversation, we extracted features from the wavefiles and the transcript, as described in the next section. We then trained six separate binary classifiers (for each gender for the 3 tasks), as described in Section 5.

4 Feature Extraction

In selecting features we drew on previous research on the use of relatively simple surface features that cue social meaning, described in the next sections.

Each date was represented by the two 4-minute wavefiles, one from the recorder worn by each speaker, and a single transcription. Because of the very high level of noise, the speaker wearing the recorder was much clearer on his/her own recording, and so we extracted the acoustic features for each speaker from their own microphone (except for the 130 dates for which we only had one audio file). All lexical and discourse features were extracted from the transcripts.

All features describe the speaker of the conversation side being labeled for style. The features for a conversation side thus indicate whether a speaker who talks a lot, laughs, is more disfluent, has higher F0, etc., is more or less likely to be considered flirtatious, friendly, or awkward by the interlocutor. We

also computed the same features for the *alter* interlocutor. *Alter* features thus indicate the conversational behavior of the speaker talking with an interlocutor they considered to be flirtatious, friendly, or awkward.

4.1 Prosodic Features

F0 and RMS amplitude features were extracted using Praat scripts (Boersma and Weenink, 2005). Since the start and end of each turn were time-marked by hand, each feature was easily extracted over a turn, and then averages and standard deviations were taken over the turns in an entire conversation side. Thus the feature F0 MIN for a conversation side was computed by taking the F0 min of each turn in that conversation side (not counting zero values of F0), and then averaging these values over all turns in the side. F0 MIN SD is the standard deviation across turns of this same measure.

Note that we coded four measures of f0 variation, not knowing in advance which one was likely to be the most useful: F0 MEAN SD is the deviation across turns from the global F0 mean for the conversation side, measuring how variable the speakers mean f0 is across turns. F0 SD is the standard deviation within a turn for the f0 mean, and then averaged over turns, hence measures how variable the speakers f0 is within a turn. F0 SD SD measures how much the within-turn f0 variance varies from turn to turn, and hence is another measure of cross-turn f0 variation. PITCH RANGE SD measures how much the speakers pitch range varies from turn to turn, and hence is another measure of cross-turn f0 variation.

F0 MIN	minimum (non-zero) F0 per turn, averaged over turns
F0 MIN SD	standard deviation from F0 min
F0 MAX	maximum F0 per turn, averaged over turns
F0 MAX SD	standard deviation from F0 max
F0 MEAN	mean F0 per turn, averaged over turns
F0 MEAN SD	standard deviation (across turns) from F0 mean
F0 SD	standard deviation (within a turn) from F0 mean, averaged over turns
F0 SD SD	standard deviation from the f0 sd
PITCH RANGE	f0 max - f0 min per turn, averaged over turns
PITCH RANGE SD	standard deviation from mean pitch range
RMS MIN	minimum amplitude per turn, averaged over turns
RMS MIN SD	standard deviation from RMS min
RMS MAX	maximum amplitude per turn, averaged over turns
RMS MAX SD	standard deviation from RMS max
RMS MEAN	mean amplitude per turn, averaged over turns
RMS MEAN SD	standard deviation from RMS mean
TURN DUR	duration of turn in seconds, averaged over turns
TIME	total time for a speaker for a conversation side, in seconds
RATE OF SPEECH	number of words in turn divided by duration of turn in seconds, averaged over turns

Table 1: Prosodic features for each conversation side, extracted using Praat from the hand-segmented turns of each side.

4.2 Lexical Features

Lexical features have been widely explored in the psychological and computational literature. For these features we drew mainly on the LIWC lexicons of Pennebaker et al. (2007), the standard for social psychological analysis of lexical features. From the large variety of lexical categories in LIWC we selected ten that the previous work of Mairesse et al. (2007) had found to be very significant in detecting personality-related features. The 10 LIWC features we used were *Anger, Assent, Ingest, Insight, Negemotion, Sexual, Swear, I, We,* and *You*. We also added two new lexical features, "past tense auxiliary", a heuristic for automatically detecting narrative or story-telling behavior, and *Metadate*, for discussion about the speed-date itself. The features are summarized in Table 2.

4.3 Dialogue Act and Adjacency Pair Features

A number of discourse features were extracted, drawing from the conversation analysis, disfluency and dialog act literature (Sacks et al., 1974; Jurafsky et al., 1998; Jurafsky, 2001). While discourse features are clearly important for extracting social meaning, previous work on social meaning has met with less success in use of such features (with the exception of the 'critical segments' work of (Enos et al., 2007)), presumably because discourse fea-

TOTAL WORDS	total number of words
PAST TENSE	uses of past tense auxiliaries *was, were, had*
METADATE	*horn, date, bell, survey, speed, form, questionnaire, rushed, study, research*
YOU	*you, you'd, you'll, your, you're, yours, you've* (not counting *you know*)
WE	*lets, let's, our, ours, ourselves, us, we, we'd, we'll, we're, we've*
I	*I'd, I'll, I'm, I've, me, mine, my, myself* (not counting *I mean*)
ASSENT	*yeah, okay, cool, yes, awesome, absolutely, agree*
SWEAR	*hell, sucks, damn, crap, shit, screw, heck, fuck**
INSIGHT	*think*/thought, feel*/felt, find/found, understand*, figure*, idea*, imagine, wonder*
ANGER	*hate/hated, hell, ridiculous*, stupid, kill*, screwed, blame, sucks, mad, bother, shit*
NEGEMOTION	*bad, weird, hate, crazy, problem*, difficult, tough, awkward, boring, wrong, sad, worry,*
SEXUAL	*love*, passion*, loves, virgin, sex, screw*
INGEST	*food, eat*, water, bar/bars, drink*, cook*, dinner, coffee, wine, beer, restaurant, lunch, dish*

Table 2: Lexical features. Each feature value is a total count of the words in that class for each conversation side; asterisks indicate that suffixed forms were included (e.g., *love, loves, loving*). All except the first three are from LIWC (Pennebaker et al., 2007) (modified slightly, for example by removing *you know* and *I mean*). The last five classes include more words in addition to those shown.

tures are expensive to hand-label and hard to automatically extract. We chose a suggestive discourse features that we felt might still be automatically extracted.

Four particular dialog acts were chosen as shown in Table 3. *Backchannels* (or continuers) and *appreciations* (a continuer expressing positive affect) were coded by hand-built regular expressions. The regular expressions were based on analysis of the backchannels and appreciations in the hand-labeled Switchboard corpus of dialog acts (Jurafsky et al., 1997). Questions were coded simply by the presence of question marks.

Finally, *repair questions* (also called NTRIs; next turn repair indicators) are turns in which a speaker signals lack of hearing or understanding (Schegloff et al., 1977). To detect these, we used a simple heuristic: the presence of 'Excuse me' or 'Wait', as in the following example:

FEMALE: Okay. Are you excited about that?
MALE: Excuse me?

A *collaborative completion* is a turn where a speaker completes the utterance begun by the alter (Lerner, 1991; Lerner, 1996). Our heuristic for identifying collaborative completions was to select sentences for which the first word of the speaker was extremely predictable from the last two words of the previous speaker. We trained a word trigram model[1] and used it to compute the probability p of the first word of a speaker's turn given the last two words of the interlocutor's turn. We arbitrarily chose the threshold .01, labeling all turns for which $p > .01$ as collaborative completions and used the total number of collaborative completions in a conversation side as our variable. This simple heuristic was errorful, but did tend to find completions beginning with *and* or *or* (1 below) and wh-questions followed by an NP or PP phrase that is grammatically coherent with the end of the question (2 and 3):

(1) FEMALE: The driving range.
(1) MALE: And the tennis court, too.

(2) MALE: What year did you graduate?
(2) FEMALE: From high school?

(3) FEMALE: What department are you in?
(3) MALE: The business school.

We also marked aspects of the *preference structure* of language. A *dispreferred* action is one in which a speaker avoids the face-threat to the interlocutor that would be caused by, e.g., refusing a request or not answering a question, by using specific strategies such as the use of *well*, hesitations, or restarts (Schegloff et al., 1977; Pomerantz, 1984).

Finally, we included the number of instances of laughter for the side, as well as the total number of turns a speaker took.

4.4 Disfluency Features

A second group of discourse features relating to repair, disfluency, and speaker overlap are summarized

[1] interpolated, with Good Turing smoothing, trained on the Treebank 3 Switchboard transcripts after stripping punctuation.

BACKCHANNELS	number of backchannel utterances in side (*Uh-huh.*, *Yeah.*, *Right.*, *Oh, okay.*)
APPRECIATIONS	number of appreciations in side (*Wow*, *That's true*, *Oh, great*)
QUESTIONS	number of questions in side
NTRI	repair question (Next Turn Repair Indicator) (*Wait*, *Excuse me*)
COMPLETION	(an approximation to) utterances that were 'collaborative completions'
DISPREFERRED	(an approximation to) dispreferred responses, beginning with discourse marker *well*
LAUGH	number of instances of laughter in side
TURNS	total number of turns in side

Table 3: Dialog act/adjacency pair features.

in Table 4. Filled pauses (*um*, *uh*) were coded by

UH/UM	total number of filled pauses (*uh* or *um*) in conversation side
RESTART	total number of disfluent restarts in conversation side
OVERLAP	number of turns in side which the two speakers overlapped

Table 4: Disfluency features

regular expressions (the transcribers had been instructed to transcribe all filled pauses). Restarts are a type of repair in which speakers begin a phrase, break off, and then restart the syntactic phrase. The following example shows a restart; the speaker starts a sentence *Uh, I* and then restarts, *There's a group...*:

Uh, I–there's a group of us that came in–

Overlaps are cases in which both speakers were talking at the same time, and were marked by the transcribers in the transcripts:

MALE:	But-and also obviously–
FEMALE:	It sounds bigger.
MALE:	–people in the CS school are not quite as social in general as other–

5 Classifier Training

Before performing the classification task, we preprocessed the data in two ways. First, we standardized all the variables to have zero mean and unit variance. We did this to avoid imposing a prior on any of the features based on their numerical values.[2] Second, we removed features correlated greater than .7. One goal of removing correlated features was to remove as much colinearity as possible from the regression so that the regression weights could be ranked for their importance in the classification. In addition, we hoped to improve classification because a large number of features require more training examples (Ng, 2004). For example for male flirt we removed *f0 range* (highly correlated with *f0 max*), *f0 min sd* (highly correlated with *f0 min*), and *Swear* (highly correlated with *Anger*).

For each classification task due to the small amounts of data we performed k-fold cross validation to learn and evaluate our models. We used a variant of k-fold cross validation with five folds where three folds are used for training, one fold is used for validation, and one fold is used as a test set. This test fold is not used in any training step. This yields a datasplit of 60% for training, 20% for validation, and 20% for testing, or 120 training examples, 40 validation examples, and 40 test examples. To ensure that we were not learning something specific to our data split, we randomized our data ordering and repeated the k-fold cross validation variant 25 times.

We used regularized logistic regression for classification. Recall that in logistic regression we train a vector of feature weights $\theta \in \mathbb{R}^n$ so as to make the following classification of some output variable y for an input observation x:[3]

$$p(y|x;\theta) = \frac{1}{1+\exp(-\theta^T x)} \quad (1)$$

In regularized logistic regression we find the

[2]Consider a feature A with mean 100 and a feature B with mean .1 where A and B are correlated with the output. Since regularization favors small weights there is a bias to put weight on feature A because intuitively the weight on feature B would need to be 1000 times larger to carry the same effect. This argument holds similarly for the reduction to unit variance.

[3]Where n is the number of features plus 1 for the intercept.

weights θ which maximize the following optimization problem:

$$\underset{\theta}{\operatorname{argmax}} \sum_i \log p(y^i|x^i;\theta) - \alpha * R(\theta) \quad (2)$$

$R(\theta)$ is a regularization term used to penalize large weights. We chose $R(\theta)$, the regularization function, to be the L_1 norm of θ. That is, $R(\theta) = ||\theta||_1 = \sum_{i=1}^{n} |\theta_i|$.

In our case, given the training set S_{train}, test set S_{test}, and validation set S_{val}, we trained the weights θ as follows:

$$\underset{\alpha}{\operatorname{argmax}} \operatorname{accuracy}(\theta_\alpha, S_{\text{val}}) \quad (3)$$

where for a given sparsity parameter α

$$\theta_\alpha = \underset{\theta}{\operatorname{argmax}} \sum_i \log p(y^i|x^i;\theta) - \alpha * R(\theta) \quad (4)$$

We chose L_1-regularization because the number of training examples to learn well grows logarithmically with the number of input variables (Ng, 2004), and to achieve a sparse activation of our features to find only the most salient explanatory variables. This choice of regularization was made to avoid the problems that often plague supervised learning in situations with large number of features but only a small number of examples. The search space over the sparsity parameter α is bounded around an expected sparsity to prevent overfitting.

Finally, to evaluate our model on the learned α and θ_α we used the features X of the test set S_{test} to compute the predicted outputs Y using the logistic regression model. Accuracy is simply computed as the percent of correct predictions.

To avoid any data ordering bias, we calculated a θ_α for each randomized run. The output of the runs was a vector of weights for each feature. We kept any feature if the median of its weight vector was nonzero.[4] A sample boxplot for the highest weighted ego features for predicting male flirt can be found in Figure 1.

[4]We also performed a t-test to find salient feature values significantly different than zero; the non-zero median method turned out to be a more conservative measure in practice (intuitively, because L_1 normed regression pushes weights to 0).

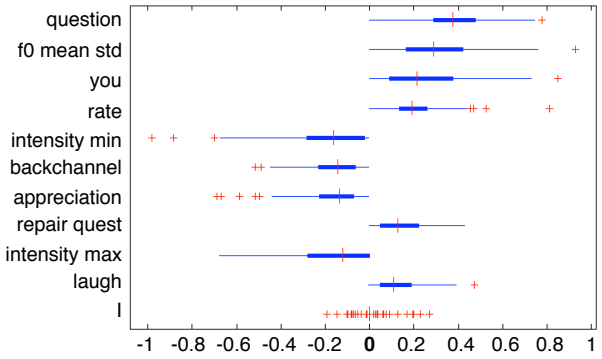


Figure 1: An illustrative boxplot for flirtation in men showing the 10 most significant features and one not significant ('I'). Shown are median values (central red line), first quartile, third quartile, outliers (red X's) and interquartile range (filled box).

6 Results

Results for the 6 binary classifiers are presented in Table 5.

	Awk		Flirt		Friendly	
	M	F	M	F	M	F
Speaker	63%	51%	67%	60%	72%	68%
+other	64%	64%	71%	60%	73%	75%

Table 5: Accuracy of binary classification of each conversation side, where chance is 50%. The first row uses features only from the single speaker; the second adds all the features from the interlocutor as well. These accuracy results were aggregated from 25 randomized runs of 5-fold cross validation.

The first row shows results using features extracted from the speaker being labeled. Here, all conversational styles are easiest to detect in men.

The second row of table 5 shows the accuracy when using features from both speakers. Not surprisingly, adding information about the interlocutor tends to improve classification, and especially for women, suggesting that male speaking has greater sway over perceptions of conversational style. We discuss below the role of these features.

We first considered the features that helped classification when considering only the ego (i.e., the results in the first row of Table 5). Table 6 shows feature weights for the features (features were normed so weights are comparable), and is summarized in the following paragraphs:

- Men who are labeled as friendly use *you*, col-

MALE FRIENDLY		MALE FLIRT	
backchannel	-0.737	question	0.376
you	0.631	f0 mean sd	0.288
intensity min sd	0.552	you	0.214
f0 sd sd	-0.446	rate	0.190
intensity min	-0.445	intensity min	-0.163
completion	0.337	backchannel	-0.142
time	-0.270	appreciation	-0.136
Insight	-0.249	repair question	0.128
f0 min	-0.226	intensity max	-0.121
intensity max	-0.221	laugh	0.107
overlap	0.213	time	-0.092
laugh	0.192	overlap	-0.090
turn dur	-0.059	f0 min	0.089
Sexual	0.059	Sexual	0.082
appreciation	-0.054	Negemo	0.075
Anger	-0.051	metadate	-0.041
FEMALE FRIENDLY		**FEMALE FLIRT**	
intensity min sd	0.420	f0 max	0.475
intensity max sd	-0.367	rate	0.346
completion	0.276	intensity min sd	0.269
repair question	0.255	f0 mean sd	0.21
appreciation	0.253	Swear	0.156
f0 max	0.233	question	-0.153
Swear	-0.194	Assent	-0.127
wordcount	0.165	f0 min	-0.111
restart	0.172	intensity max	0.092
uh	0.241	I	0.073
I	0.111	metadate	-0.071
past	-0.060	wordcount	0.065
laugh	0.048	laugh	0.054
Negemotion	-0.021	restart	0.046
intensity min	-0.02	overlap	-0.036
Ingest	-0.017	f0 sd sd	-0.025
Assent	0.0087	Ingest	-0.024
f0 max sd	0.0089		
MALE AWK			
restart	0.502	completion	-0.141
f0 sd sd	0.371	intensity max	-0.135
appreciation	-0.354	f0 mean sd	-0.091
turns	-0.292	Ingest	-0.079
uh	0.270	Anger	0.075
you	-0.210	repair question	-0.067
overlap	-0.190	Insight	-0.056
past	-0.175	rate	0.049
intensity min sd	-0.173		

Table 6: Feature weights (median weights of the randomized runs) for the non-zero predictors for each classifier. Since our accuracy for detecting awkwardness in women based solely on ego features is so close to chance, we didn't analyze the awkwardness features for women here.

laborative completions, laugh, overlap, but don't backchannel or use appreciations. Their utterances are shorter (in seconds and words) and they are quieter and their (minimum) pitch is lower and somewhat less variable.

• Women labeled as friendly have more collaborative completions, repair questions, laughter, and appreciations. They use more words overall, and use *I* more often. They are more disfluent (both restarts and *uh*) but less likely to swear. Prosodically their f0 is higher, and there seems to be some pattern involving quiet speech; more variation in intensity minimum than intensity max.

• Men who are labeled as flirting ask more questions, including repair questions, and use *you*. They don't use backchannels or appreciations, or overlap as much. They laugh more, and use more sexual and negative emotional words. Prosodically they speak faster, with higher and more variable pitch, but quieter (lower intensity max).

• The strongest features for women who are labeled as flirting are prosodic; they speak faster and louder with higher and more variable pitch. They also use more words in general, swear more, don't ask questions or use Assent, use more I, laugh more, and are somewhat more disfluent (restarts).

• Men who are labeled as awkward are more disfluent, with increased restarts and filled pauses (*uh* and *um*). They are also not 'collaborative' conversationalists; they don't use appreciations, repair questions, collaborative completions, past-tense, or *you*, take fewer turns overall, and don't overlap. Prosodically the awkward labels are hard to characterize; there is both an increase in pitch variation (f0 sd sd) and a decrease (f0 mean sd). They don't seem to get quite as loud (intensity max).

The previous analysis showed what features of the ego help in classification. We next asked about features of the alter, based on the results using both ego and alter features in the second row of Table 5. Here we are asking about the linguistic behaviors of a speaker who describes the interlocutor as flirting, friendly, or awkward.

While we don't show these values in a table, we offer here an overview of their tendencies. For example for women who labeled their male interlocutors as friendly, the women got much quieter, used 'well' much more, laughed, asked more

repair questions, used collaborative completions, and backchanneled more. When a man labeled a woman as friendly, he used an expanded intensity range (quieter intensity min, louder intensity max). laughed more, used more sexual terms, used less negative emotional terms, and overlapped more.

When women labeled their male interlocutor as flirting, the women used many more repair questions, laughed more, and got quieter (lower intensity min). By contrast, when a man said his female interlocutor was flirting, he used more Insight and Anger words, and raised his pitch.

When women labeled their male interlocutor as awkward, the women asked a lot of questions, used *well*, were disfluent (restarts), had a diminished pitch range, and didn't use *I*. In listening to some of these conversations, it was clear that the conversation lagged repeatedly, and the women used questions at these points to restart the conversations.

7 Discussion

The results presented here should be regarded with some caution. The sample is not a random sample of English speakers or American adults, and speed dating is not a natural context for expressing every conversational style. Therefore, a wider array of studies across populations and genres would be required before a more general theory of conversational styles is established.

On the other hand, the presented results may under-reflect the relations being captured. The quality of recordings and coarse granularity (1 second) of the time-stamps likely cloud the relations, and as the data is cleaned and improved, we expect the associations to only grow stronger.

Caveats aside, we believe the evidence indicates that the perception of several types of conversational style have relatively clear signals across genders, but with some additional gender contextualization.

Both genders convey flirtation by laughing more, speaking faster, and using higher and more variable pitch. Both genders convey friendliness by laughing more, and using collaborative completions.

However, we do find gender differences; men asl more questions when (labeled as) flirting, women ask fewer. Men labeled as flirting are softer, but women labeled as flirting are louder. Women flirting swear more, while men are more likely to use sexual vocabulary. Gender differences exist as well for the other variables. Men labeled as friendly use *you* while women labeled as friendly use *I*. Friendly women are very disfluent; friendly men are not.

While the features for friendly and flirtatious speech overlap, there are clear differences. Men speaker faster and with higher f0 (min) in flirtatious speech, but not faster and with lower f0 (min) in friendly speech. For men, flirtatious speech involves more questions and repair questions, while friendly speech does not. For women, friendly speech is more disfluent than flirtatious speech, and has more collaborative style (completions, repair questions, appreciations).

We also seem to see a model of *collaborative conversational style* (probably related to the *collaborative floor* of Edelsky (1981) and Coates (1996)), cued by the use of more collaborative completions, repair questions and other questions, *you*, and laughter. These collaborative techniques were used by both women and men who were labeled as friendly, and occurred less with men labeled as awkward. Women themselves displayed more of this collaborative conversational style when they labeled the men as friendly. For women only, collaborative style included appreciations; while for men only, collaborative style included overlaps.

In addition to these implications for social science, our work has implications for the extraction of meaning in general. A key focus of our work was on ways to extract useful dialog act and disfluency features (repair questions, backchannels, appreciations, restarts, dispreferreds) with very shallow methods. These features were indeed extractable and proved to be useful features in classification.

We are currently extending these results to predict date outcomes including 'liking', extending work such as Madan and Pentland (2006).

Acknowledgments

Thanks to three anonymous reviewers, Sonal Nalkur and Tanzeem Choudhury for assistance and advice on data collection, Sandy Pentland for a helpful discussion about feature extraction, and to Google for gift funding.

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 *INTERSPEECH-02*.

P. Boersma and D. Weenink. 2005. Praat: doing phonetics by computer (version 4.3.14). [Computer program]. Retrieved May 26, 2005, from `http://www.praat.org/`.

S. Brave, C. Nass, and K. Hutchinson. 2005. Computers that care: Investigating the effects of orientation of emotion exhibited by an embodied conversational agent. *International Journal of Human-Computer Studies*, 62(2):161–178.

S. E. Brennan and M. F. Schober. 2001. How listeners compensate for disfluencies in spontaneous speech. *Journal of Memory and Language*, 44:274–296.

S. E. Brennan and M. Williams. 1995. The feeling of another's knowing: Prosody and filled pauses as cues to listeners about the metacognitive states of speakers. *Journal of Memory and Language*, 34:383–398.

J. Coates. 1996. *Women Talk*. Blackwell.

M. A. Cohn, M. R. Mehl, and J. W. Pennebaker. 2004. Linguistic markers of psychological change surrounding September 11, 2001. *Psychological Science*, 15:687–693.

C. Edelsky. 1981. Who's got the floor? *Language in Society*, 10:383–421.

F. Enos, E. Shriberg, M. Graciarena, J. Hirschberg, and A. Stolcke. 2007. Detecting Deception Using Critical Segments. In *INTERSPEECH-07*.

D. Jurafsky, E. Shriberg, and D. Biasca. 1997. Switchboard SWBD-DAMSL Labeling Project Coder's Manual, Draft 13. Technical Report 97-02, University of Colorado Institute of Cognitive Science.

D. Jurafsky, E. Shriberg, B. Fox, and T. Curl. 1998. Lexical, prosodic, and syntactic cues for dialog acts. In *Proceedings, COLING-ACL Workshop on Discourse Relations and Discourse Markers*, pages 114–120.

D. Jurafsky. 2001. Pragmatics and computational linguistics. In L. R. Horn and G. Ward, editors, *Handbook of Pragmatics*. Blackwell.

C. M. Lee and S. S. Narayanan. 2002. Combining acoustic and language information for emotion recognition. In *ICSLP-02*, pages 873–876, Denver, CO.

G. H. Lerner. 1991. On the syntax of sentences-in-progress. *Language in Society*, 20(3):441–458.

G. H. Lerner. 1996. On the "semi-permeable" character of grammatical units in conversation: Conditional entry into the turn space of another speaker. In E. Ochs, E. A. Schegloff, and S. A. Thompson, editors, *Interaction and Grammar*, pages 238–276. Cambridge University Press.

J. Liscombe, J. Venditti, and J. Hirschberg. 2003. Classifying Subject Ratings of Emotional Speech Using Acoustic Features. In *INTERSPEECH-03*.

A. Madan and A. Pentland. 2006. Vibefones: Socially aware mobile phones. In *Tenth IEEE International Symposium on Wearable Computers*.

A. Madan, R. Caneel, and A. Pentland. 2005. Voices of attraction. Presented at Augmented Cognition, HCI 2005, Las Vegas.

F. Mairesse and M. Walker. 2008. Trainable generation of big-five personality styles through data-driven parameter estimation. In *ACL-08*, Columbus.

F. Mairesse, M. Walker, M. Mehl, and R. Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *Journal of Artificial Intelligence Research (JAIR)*, 30:457–500.

C. Nass and S. Brave. 2005. *Wired for speech: How voice activates and advances the human-computer relationship*. MIT Press, Cambridge, MA.

M. L. Newman, J. W. Pennebaker, D. S. Berry, and J. M. Richards. 2003. Lying words: Predicting deception from linguistic style. *Personality and Social Psychology Bulletin*, 29:665–675.

A. Y. Ng. 2004. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *ICML 2004*.

J. W. Pennebaker and T. C. Lay. 2002. Language use and personality during crises: Analyses of Mayor Rudolph Giuliani's press conferences. *Journal of Research in Personality*, 36:271–282.

J. W. Pennebaker, R. Booth, and M. Francis. 2007. Linguistic inquiry and word count: LIWC2007 operator's manual. Technical report, University of Texas.

A. Pentland. 2005. Socially aware computation and communication. *Computer*, pages 63–70.

A. M. Pomerantz. 1984. Agreeing and disagreeing with assessment: Some features of preferred/dispreferred turn shapes. In J. M. Atkinson and J. Heritage, editors, *Structure of Social Action: Studies in Conversation Analysis*. Cambridge University Press.

A. Rosenberg and J. Hirschberg. 2005. Acoustic/prosodic and lexical correlates of charismatic speech. In *EUROSPEECH-05*, pages 513–516, Lisbon, Portugal.

S. S. Rude, E. M. Gortner, and J. W. Pennebaker. 2004. Language use of depressed and depression-vulnerable college students. *Cognition and Emotion*, 18:1121–1133.

H. Sacks, E. A. Schegloff, and G. Jefferson. 1974. A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735.

E. A. Schegloff, G. Jefferson, and H. Sacks. 1977. The preference for self-correction in the organization of repair in conversation. *Language*, 53:361–382.

Linear Complexity Context-Free Parsing Pipelines via Chart Constraints

Brian Roark and Kristy Hollingshead
Center for Spoken Language Understanding
Division of Biomedical Computer Science
Oregon Health & Science University
{roark,hollingk}@cslu.ogi.edu

Abstract

In this paper, we extend methods from Roark and Hollingshead (2008) for reducing the worst-case complexity of a context-free parsing pipeline via hard constraints derived from finite-state tagging pre-processing. Methods from our previous paper achieved quadratic worst-case complexity. We prove here that alternate methods for choosing constraints can achieve either linear or $O(N\log^2 N)$ complexity. These worst-case bounds on processing are demonstrated to be achieved without reducing the parsing accuracy, in fact in some cases improving the accuracy. The new methods achieve observed performance comparable to the previously published quadratic complexity method. Finally, we demonstrate improved performance by combining complexity bounding methods with additional high precision constraints.

1 Introduction

Finite-state pre-processing for context-free parsing is very common as a means of reducing the amount of search required in the later stage. For example, the well-known Ratnaparkhi parser (Ratnaparkhi, 1999) used a finite-state POS-tagger and NP-chunker to reduce the search space for his Maximum Entropy parsing model, and achieved linear observed-time performance. Other recent examples of the utility of finite-state constraints for parsing pipelines include Glaysher and Moldovan (2006), Djordjevic et al. (2007), Hollingshead and Roark (2007), and Roark and Hollingshead (2008). Note that by making use of constraints derived from pre-processing, they are no longer performing full exact inference—these are approximate inference methods, as are the methods presented in this paper. Most of these parsing pipeline papers show empirically that these techniques can improve pipeline efficiency for well-known parsing tasks. In contrast, in Roark and Hollingshead (2008), we derived and applied the finite-state constraints so as to *guarantee* a reduction in the worst-case complexity of the context-free parsing pipeline from $O(N^3)$ in the length of the string N to $O(N^2)$ by closing chart cells to entries. We demonstrated the application of such constraints to the well-known Charniak parsing pipeline (Charniak, 2000), which resulted in no accuracy loss when the constraints were applied.

While it is important to demonstrate that these sorts of complexity-reducing chart constraints do not interfere with the operation of high-accuracy, state-of-the-art parsing approaches, existing pruning techniques used within such parsers can obscure the impact of these constraints on search. For example, using the default search parameterization of the Charniak parser, the Roark and Hollingshead (2008) results demonstrated no parser speedup using the techniques, rather an *accuracy* improvement, which we attributed to a better use of the amount of search permitted by that default parameterization. We only demonstrated efficiency improvements by reducing the amount of search via the Charniak search parameterization. There we showed a nice speedup of the parser versus the default, while maintaining accuracy levels. However, internal heuristics of the Charniak search, such as attention shifting (Blaheta and Charniak, 1999; Hall and Johnson, 2004), can make this accuracy/efficiency tradeoff somewhat difficult to interpret.

Furthermore, one might ask whether $O(N^2)$ complexity is as good as can be achieved through the paradigm of using finite-state constraints to close chart cells. What methods of constraint would be required to achieve $O(N \log N)$ or linear complex-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 647–655,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

ity? Would such constraints degrade performance, or can the finite-state models be applied with sufficient precision to allow for such constraints without significant loss of accuracy?

In this paper, we adopt the same paradigm pursued in Roark and Hollingshead (2008), but apply it to an exact inference CYK parser (Cocke and Schwartz, 1970; Younger, 1967; Kasami, 1965). We demonstrate that imposing constraints sufficient to achieve quadratic complexity in fact yields observed linear parsing time, suggesting that tighter complexity bounds are possible. We prove that a different method of imposing constraints on words beginning or ending multi-word constituents can give $O(N\log^2 N)$ or $O(N)$ worst-case complexity, and we empirically evaluate the impact of such an approach.

The rest of the paper is structured as follows. We begin with a summary of the chart cell constraint techniques from Roark and Hollingshead (2008), and some initial empirical trials applying these techniques to an exact inference CYK parser. Complexity bounding approaches are contrasted (and combined) with high precision constraint selection methods from that paper. We then present a new approach to making use of the same sort of finite-state tagger output to achieve linear or $N\log^2 N$ complexity. This is followed with an empirical validation of the new approach.

2 Background: Chart Cell Constraints

The basic algorithm from Roark and Hollingshead (2008) is as follows. Let B be the set of words in a string $w_1 \dots w_k$ that begin a multi-word constituent, and let E be the set of words in the string that end a multi-word constituent. For chart parsing with, say, the CYK algorithm, cells in the chart represent substrings $w_i \dots w_j$ of the string, and can be indexed with (i, j), the beginning and ending words of the substring. If $w_i \notin B$, then we can *close* any cell (i, j) where $i < j$, i.e., no complete constituents need be stored in that cell. Similarly, if $w_j \notin E$, then we can close any cell (i, j) where $i < j$. A discriminatively trained finite-state tagger can be used to classify words as being in or out of these sets with relatively high tagging accuracy, around 97% for both sets (B and E). The output of the tagger is then used to close cells, thus reducing the work for the chart parser.

An important caveat must be made about these closed cells, related to *incomplete* constituents. For simplicity of exposition, we will describe incomplete constituents in terms of factored categories in a Chomsky Normal Form grammar, e.g., the new non-terminal $Z{:}X{+}W$ that results when the ternary rule production $Z \rightarrow Y \;\; X \;\; W$ is factored into the two binary productions $Z \rightarrow Y \;\; Z{:}X{+}W$ and $Z{:}X{+}W \rightarrow X \;\; W$. A factored category such as $Z{:}X{+}W$ should be permitted in cell (i, j) if $w_j \in E$, even if $w_i \notin B$, because the category could subsequently combine with an Y category to create a Z constituent that begins at some word $w_p \in B$. Hence there are three possible conditions for cell (i, j) in the chart:

1. $w_j \notin E$: closing the cell affects *all* constituents, both complete and incomplete
2. $w_i \notin B$ **and** $w_j \in E$: closing the cell affects only complete constituents
3. $w_i \in B$ and $w_j \in E$: cell is not closed, i.e., it is "open"

In Roark and Hollingshead (2008), we proved that, for the CYK algorithm, there is no work necessary for case 1 cells, a constant amount of work for case 2 cells, and a linear amount of work for case 3 cells. Therefore, if the number of cells allowed to fall in case 3 is linear, the overall complexity of search is $O(N^2)$.

The amount of work for each case is related to how the CYK algorithm performs its search. Each cell in the chart (i, j) represents a substring $w_i \dots w_j$, and building non-terminal categories in that cell involves combining non-terminal categories (via rules in the context-free grammar) found in cells of adjacent substrings $w_i \dots w_m$ and $w_{m+1} \dots w_j$. The length of substrings can be up to order N (length of the whole string), hence there are $O(N)$ *midpoint* words w_m in the standard algorithm, and in the case 3 cells above. This accounts for the linear amount of work for those cells. Case 2 cells have constant work because there is only one possible midpoint, and that is w_i, i.e., the first child of any incomplete constituent placed in a case 2 cell must be span 1, since $w_i \notin B$. This is a very concise recap of the proof, and we refer the reader to our previous paper for more details.

3 Constraining Exact-Inference CYK

Despite referring to the CYK algorithm in the proof, in Roark and Hollingshead (2008) we demonstrated our approach by constraining the Charniak parser (Charniak, 2000), and achieved an improvement in the accuracy/efficiency tradeoff curve. However, as mentioned earlier, the existing complicated system of search heuristics in the Charniak parser makes interpretation of the results more difficult. What can be said from the previous results is that constraining parsers in this way can improve performance of even the highest accuracy parsers. Yet those results do not provide much of an indication of how performance is impacted for general context-free inference.

For this paper, we use an exact inference (exhaustive search) CYK parser, using a simple probabilistic context-free grammar (PCFG) induced from the Penn WSJ Treebank (Marcus et al., 1993). The PCFG is transformed to Chomsky Normal Form through right-factorization, and is smoothed with a Markov (order-2) transform. Thus a production such as $Z \rightarrow Y \;\; X \;\; W \;\; V$ becomes three rules: (1) $Z \rightarrow Y \;\; Z{:}X{+}W$; (2) $Z{:}X{+}W \rightarrow X \;\; Z{:}W{+}V$; and (3) $Z{:}W{+}V \rightarrow W \;\; V$. Note that only two child categories are encoded within the new factored categories, instead of all of the remaining children as in our previous factorization example. This so-called 'Markov' grammar provides some smoothing of the PCFG; the resulting grammar is also smoothed using lower order Markov grammars.

We trained on sections 2-21 of the treebank, and all results except for the final table are on the development section (24). The final table is on the test section (23). All results report F-measure labeled bracketing accuracy for all sentences in the section.

To close cells, we use a discriminatively trained finite-state tagger to tag words as being either in B or not, and also (in a separate pass) either in E or not. Note that the reference tags for each word can be derived directly from the treebank, based on the spans of constituents beginning (or ending) at each word. Note also that these reference tags are based on a non-factored grammar.

For example, consider the chart in Figure 1 for the five symbol string "$abcde$". Each cell in the chart is labeled with the substring that the cell spans, along with the begin and end indices of the substring, e.g., $(3, 5)$ spans the third symbol to the fifth symbol: cde. If our tagger output is such that $b \notin E$ and $d \notin E$, then four cells will be closed: $(1, 2)$, $(1, 4)$, $(2, 4)$ and $(3, 4)$. The gray shaded cells in the figure have some midpoints that require no work, because they involve closed children cells.

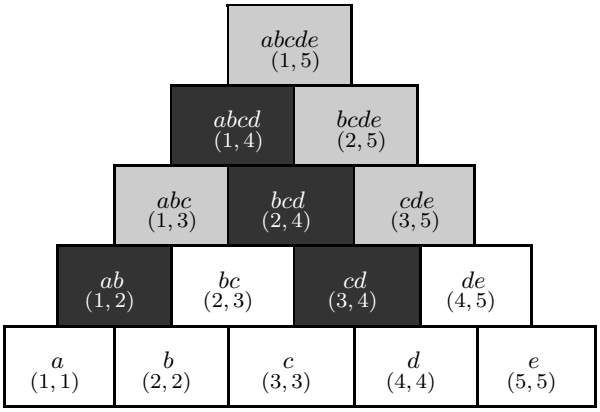


Figure 1: Fragment of a chart structure. Each cell is labeled with the substring spanned by that cell, along with the start and end word indices. Cell shading reflects $b \notin E$ and $d \notin E$ constraints: black denotes "closed" cells; white and gray are "open"; gray cells have "closed" children cells, reducing the number of midpoints requiring processing.

4 Constraint Selection

4.1 High Precision vs Complexity Bounding

The chart constraints that are extracted from the finite-state tagger come in the form of set exclusions, e.g., $d \notin E$. Rather than selecting constraints from the single, best-scoring tag sequence output by the tagger, we instead rely on the whole distribution over possible tag strings to select constraints. We have two separate tagging tasks, each with two possible tags of each word w_i in each string: (1) B or ¬B; and (2) E or ¬E, where ¬X signifies that $w_i \notin X$ for $X \in \{B, E\}$. The tagger (Hollingshead et al., 2005) uses log linear models trained with the perceptron algorithm, and derives, via the forward-backward algorithm, the posterior probability of each of the two tags at each word, so that $\Pr(\text{B}) + \Pr(\neg\text{B}) = 1$. Then, for every word w_i in the string, the tags B and E are associated with a posterior probability that gives us a score for $w_i \in B$ and $w_i \in E$. All possible set memberships $w_i \in X$ in the string can be ranked by this score. From this ranking, a decision boundary can be set, such that all word/set pairs $w_i \in B$ or $w_j \in E$ with above-threshold probability are accepted, and all pairs below threshold are excluded from the set.

The default decision boundary for this tagging

task is 0.5 posterior probability (more likely than not), and tagging performance at that threshold is good (around 97% accuracy, as mentioned previously). However, since this is a pre-processing step, we may want to reduce possible cascading errors by allowing more words into the sets B and E. In other words, we may want more precision in our set exclusion constraints. One method for this is to count the number c of word/set pairs below posterior probability of 0.5, then set the threshold so that only kc word/set pairs fall below threshold, where $0 < k \leq 1$. Note that the closer the parameter k is to 0, the fewer constraints will be applied to the chart. We refer to the resulting constraints as "high precision", since the selected constraints (set exclusions) have high precision. This technique was also used in the previous paper.

We also make use of the ranked list of word/set pairs to impose quadratic bounds on context-free parsing. Starting from the top of the list (highest posterior probability for set inclusion), word/set pairs are selected and the number of open cells (case 3 in Section 2) calculated. When the accumulated number of open cells reaches kN for sentence length N, the decision threshold is set. In such a way, there are only a linear number of open, case 3 cells, hence the parsing has quadratic worst-case complexity.

For both of these methods, the parameter k can vary, allowing for more or less set inclusion. Figure 2 shows parse time versus F-measure parse accuracy on the development set for the baseline (unconstrained) exact-inference CYK parser, and for various parameterizations of both the high precision constraints and the quadratic bound constraints. Note that accuracy actually improves with the imposition of these constraints. This is not surprising, since the finite-state tagger deriving the constraints made use of lexical information that the simple PCFG did not, hence there is complementary information improving the model. The best operating points—fast parsing and relatively high accuracy—are achieved with 90% of the high precision constraints, and $5N$ cells left open. These achieve a roughly 20 times speedup over the baseline unconstrained parser and achieve between 1.5 and 3 percent accuracy gains over the baseline.

We can get a better picture of what is going on by considering the scatter plots in Figure 3, which plot

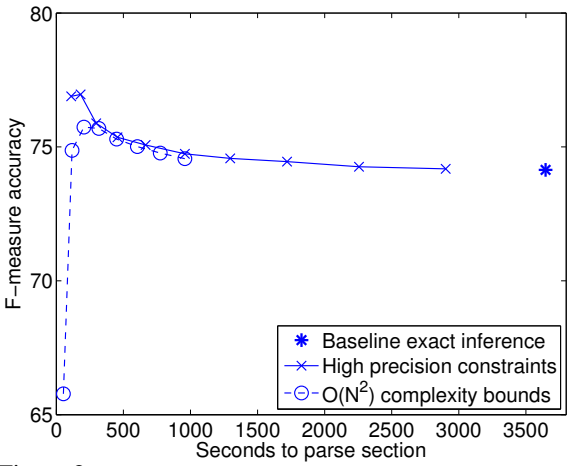


Figure 2: Time to parse (seconds) versus accuracy (F-measure) for the baseline of exact inference (no constraints) versus two methods of imposing constraints with varying parameters: (1) High precision constraints; (2) Sufficient constraints to impose $O(N^2)$ complexity (the number of open cells $\leq kN$).

each sentence according to its length versus the parsing time for that sentence at three operating points: baseline (unconstrained); high precision at 90%; and quadratic with $5N$ open cells. The top plot shows up to 120 words in the sentence, and up to 5 seconds of parsing time. The middle graph zooms in to under 1 second and up to 60 words; and the lowest graph zooms in further to under 0.1 seconds and up to 20 words. It can be seen in each graph that the unconstrained CYK parsing quickly leaves the graph via a steep cubic curve.

Three points can be taken away from these plots. First, the high precision constraints are better for the shorter strings than the quadratic bound constraints (see bottom plot); yet with the longer strings, the quadratic constraints better control parsing time than the high precision constraints (see top plot). Second, the quadratic bound constraints appear to actually result in roughly linear parsing time, not quadratic. Finally, at the "crossover" point, where quadratic constraints start out-performing the high precision constraints (roughly 40-60 words, see middle plot), there is quite high variance in high precision constraints versus the quadratic bounds: some sentences process more quickly than the quadratic bounds, some quite a bit worse. This illustrates the difference between the two methods of selecting constraints: the high precision constraints can provide very strong gains, but there is no guarantee for the worst case. In such a way, the high precision constraints are similar to other tagging-derived

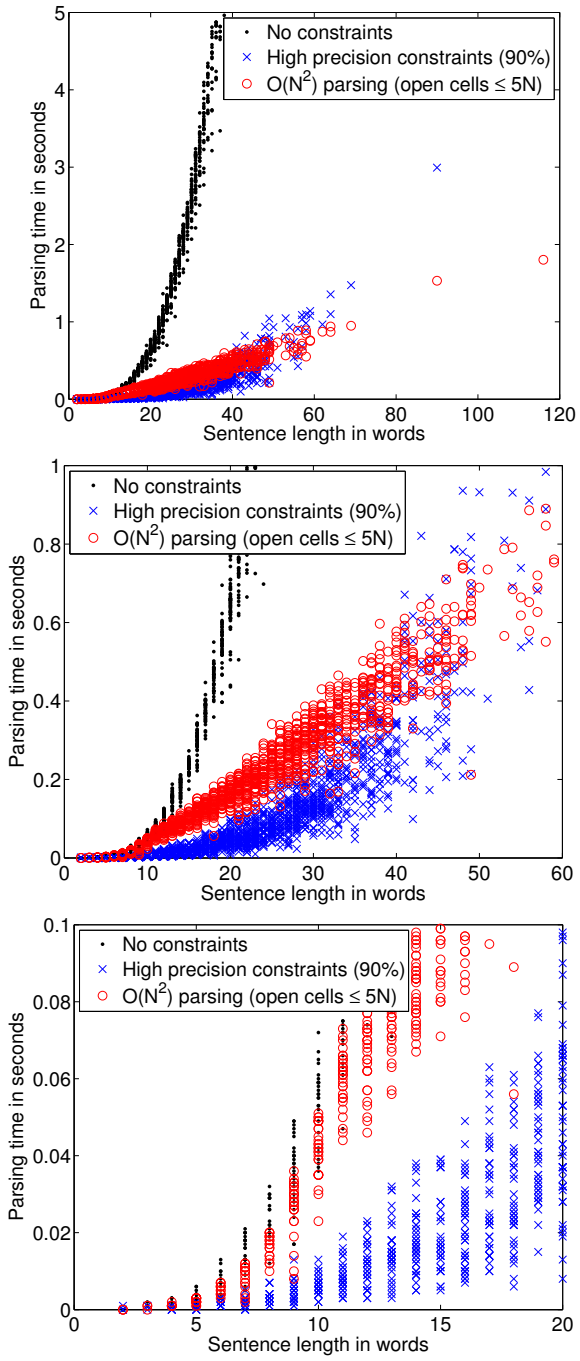


Figure 3: Scatter plots of sentence length versus parsing time for (1) baseline exact inference (no constraints); (2) high precision begin- and end-constituent constraints (90% level); and (3) $O(N^2)$ constraints (5N open cells).

constraints like POS-tags or chunks.

4.2 Combining Constraints

Depending on the length of the string, the quadratic constraints may close more or fewer chart cells than the high precision constraints—more for long strings, fewer for short strings. We can achieve worst-case bounds, along with superior typical case speedups, by combining both methods as follows: first apply the quadratic bounds; then, if there are any high precision constraints that remain unapplied, add them. Table 1 shows F-measure accuracy and parsing time (in seconds) for four trials on the development set: the baseline CYK with no constraints; high precision constraints at the 90% level; quadratic bound constraints at the $5N$ level; and a combination of the quadratic bound and high precision constraints. We can see that, indeed, the combination of the two yield speedups over both independently, with no significant drop in accuracy from the high precision constraints alone. Further results with worst-case complexity bounds will be combined with high precision constraints in this way.

Constraints	F-measure accuracy	time (seconds)
None (baseline CYK)	74.1	3646
High Precision (90%)	77.0	181
Quadratic ($5N$)	75.7	317
Quad ($5N$) + HiPrec (90%)	76.9	166

Table 1: Speed and accuracy of exact-inference CYK parser on WSJ section 24 under various constraint conditions, including combining quadratic bound constraints and high precision constraints.

The observed linear parsing time in Figure 3 with the quadratic constraints raises the following question: can we apply these constraints in a way that guarantees linear complexity? The answer is yes, and this is the subject of the next section.

5 Linear and $N\log^2 N$ Complexity Bounds

Given the two sets B and E, recall the three cases of chart cells (i, j) presented in Section 2: 1) $w_j \notin E$ (cell completely closed); 2) $w_j \in E$ and $w_i \notin B$ (cell open only for incomplete constituents); and 3) $w_i \in B$ and $w_j \in E$ (cell open for all constituents). Quadratic worst-case complexity is achieved with these sets by limiting case 3 to hold for only $O(N)$ cells—each with linear work—and the remaining $O(N^2)$ cells (cases 1 and 2) have none or constant work, hence overall quadratic (Roark and Hollingshead, 2008).

One might ask: why would imposing constraints to achieve a quadratic bound give us linear observed parsing time? One possibility is that the linear number of case 3 cells don't have a linear amount of work, but rather a constant bounded amount of work.

If there were a constant bounded number of midpoints, then the amount of work associated with case 3 would be linear. Note that a linear complexity bound would have to guarantee a linear number of case 2 cells as well since there is a constant amount of work associated with case 2 cells.

To provide some intuition as to why the quadratic bound method resulted in linear observed parsing time, consider again the chart structure in Figure 1. The black cells in the chart represent the cells that have been closed when $w_j \notin E$ (case 1 cells). In our example, $w_2 \notin E$ caused the cell spanning ab to be closed, and $w_4 \notin E$ caused the cells spanning $abcd$, bcd and cd to be closed. Since there is no work required for these cells, the amount of work required to parse the sentence is reduced. However, the quadratic bound does not include any potential reduced work in the remaining open cells. The gray cells in the chart are cells with a reduced number of possible midpoints, as effected by the closed cells in the chart. For example, categories populating the cell spanning abc in position $(1,3)$ can be built in two ways: either by combining entries in cell $(1,1)$ with entries in $(2,3)$ at midpoint $m = 1$; or by combining entries in $(1,2)$ and $(3,3)$ at midpoint $m = 2$. However, cell $(1,2)$ is closed, hence there is only one midpoint at which $(1,3)$ can be built ($m = 1$). Thus the amount of work to parse the sentence will be less than the worst-case quadratic bound based on this processing savings in open cells.

While imposition of the quadratic bound may have resulted (fortuitously) in constant bounded work for case 3 cells and a linear number of case 2 cells, there is no guarantee that this will be the case. One method to guarantee that both conditions are met is the following: if $|E| \leq k$ for some constant k, then both conditions will be met and parsing complexity will be linear. We prove here that constraining E to contain a constant number of words results in linear complexity.

Lemma 1: *If $|E| \leq k$ for some k, then the amount of work for any cell is bounded by ck for some constant c (grammar constant).*

Proof: Recall from Section 2 that for each cell (i,j), there are $j-i$ midpoints m that require combining entries in cells (i,m) and $(m+1,j)$ to create entries in cell (i,j). If $m > i$, then cell (i,m) is empty unless $w_m \in E$. If cell (i,m) is empty, there is no work to be done at that midpoint. If $|E| \leq k$, then there are a maximum of k midpoints for any cell, hence the amount of work is bounded by ck for some constant c.□

Lemma 2: *If $|E| \leq k$ for some k, then the number of cells (i,j) such that $w_j \in E$ is no more than kN where N is the length of the string.*

Proof: For a string of length N, each word w_j in the string has at most N cells such that w_j is the last word in the substring spanned by that cell, since each such cell must begin with a distinct word w_i in the string where $i \leq j$, of which there are at most N. Therefore, if $|E| \leq k$ for some k, then the number of cells (i,j) such that $w_j \in E$ would be no more than kN.□

Theorem: *If $|E| \leq k$, then the parsing complexity is $O(k^2N)$.*

Proof: As stated earlier, each cell (i,j) falls in one of three cases: 1) $w_j \notin E$; 2) $w_j \in E$ and $w_i \notin B$; and 3) $w_i \in B$ and $w_j \in E$. Case 1 cells are completely closed, there is no work to be done in those cells. By Lemma 2, there are at maximum kN cells that fall in either case 2 or case 3. By Lemma 1, the amount of work for each of these cells is bounded by ck for some constant c. Therefore, the theorem is proved.□

If $|E| \leq k$ for a constant k, the theorem proves the complexity will be $O(N)$. If $|E| \leq k \log N$, then parsing complexity will be $O(N\log^2 N)$. Figure 4 shows sentence length versus parsing time under three different conditions[1]: baseline (unconstrained); $O(N\log^2 N)$ at $|E| \leq 3 \log N$; and linear at $|E| \leq 16$. The bottom graph zooms in to demonstrate that the $O(N\log^2 N)$ constraints can outperform the linear constraints for shorter strings (see around 20 words). As the length of the string increases, though, the performance lines cross, and the linear constraints demonstrate higher efficiency for the longer strings, as expected.

Unlike the method for imposing quadratic bounds, this method only makes use of set E, not B. To select the constraints, we rank the word/E posterior probabilities, and choose the top k (either constant or scaled with a $\log N$ factor); the rest of the words fall outside of the set. In this approach,

[1]Selection of these particular operating points for the $N\log^2 N$ and linear methods is discussed in Section 6.

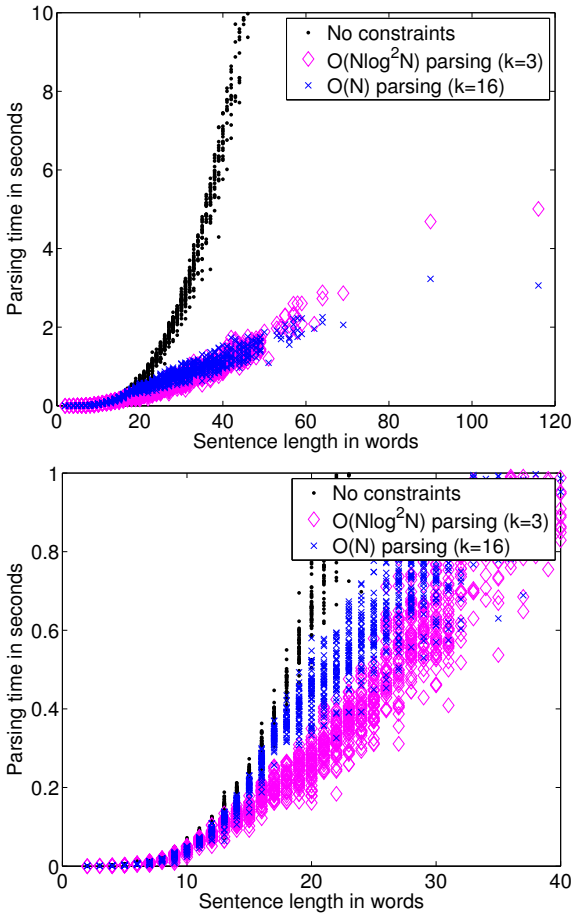


Figure 4: Scatter plots of sentence length versus parsing time for (1) baseline exact inference (no constraints); (2) $O(N\log^2 N)$ constraints; and (3) $O(N)$ constraints.

every word falls in the B set, hence no constraints on words beginning multi-word constituents are imposed.

6 Results

Figure 5 plots F-measure accuracy versus time to parse the development set for four methods of imposing constraints: the previously plotted high precision and quadratic bound constraints, along with $O(N\log^2 N)$ and linear bound constraints using methods described in this paper. All methods are employed at various parameterizations, from very lightly constrained to very heavily constrained. The complexity-bound constraints are not combined with the high-precision constraints for this plot.

As can be seen from the plot, the linear and $O(N\log^2 N)$ methods do not, as applied, achieve as favorable of an accuracy/efficiency tradeoff curve as the quadratic bound method. This is not surprising,

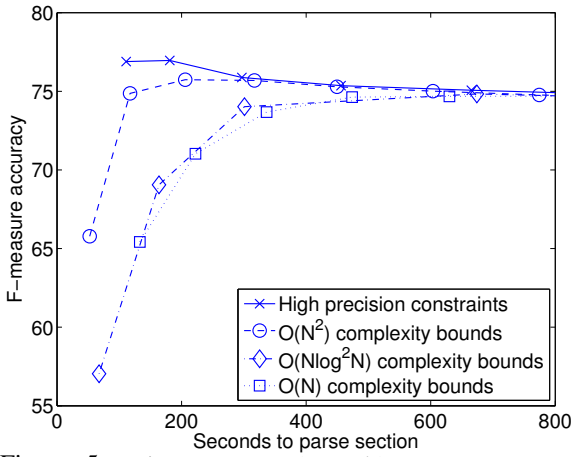


Figure 5: Time to parse (seconds) versus accuracy (F-measure) for high precision constraints of various thresholds versus three methods of imposing constraints with complexity bounds: (1) $O(N^2)$ complexity (number of open cells $\leq kN$); (2) $O(N\log^2 N)$ complexity ($|E| \leq k \log N$); and (3) linear complexity ($|E| \leq k$).

given that no words are excluded from the set B for these methods, hence far fewer constraints overall are applied with the new method than with the quadratic bound method.

Of course, the high precision constraints can be applied together with the complexity bound constraints, as described in Section 4.2. For combining complexity-bound constraints with high-precision constraints, we first chose operating points for both the linear and $O(N\log^2 N)$ complexity bound methods at the points before accuracy begins to degrade with over-constraint. For the linear complexity method, the operating point is to constrain the set size of E to a maximum of 16 members, i.e., $|E| \leq 16$. For the $N\log^2 N$ complexity method, $|E| \leq 3 \log N$.

Table 2 presents results for these operating points used in conjunction with the 90% high precision constraints. For these methods, this combination is particularly important, since it includes all of the high precision constraints from the set B, which are completely ignored by both of the new methods. We can see from the results in the table that the combination brings the new constraint methods to very similar accuracy levels as the quadratic constraints, yet with the guarantee of scaling linearly to longer and longer sentences.

The efficiency benefits of combining constraints, shown in Table 2, are relatively small here because the dataset contains mostly shorter sentences. Space

Constraints	F-measure accuracy	time (seconds)
None (baseline CYK)	74.1	3646
High Precision (90%)	77.0	181
Quad ($5N$) + HiPrec (90%)	76.9	166
$N\log^2 N$ ($3\log N$) + HP (90)	76.9	170
Linear (16) + HiPrec (90%)	76.8	167

Table 2: Speed and accuracy of exact-inference CYK parser on WSJ section 24 under various constraint conditions, including combining various complexity bound constraints and high precision constraints.

Constraints	F-measure accuracy	time (seconds)
None (baseline CYK)	73.8	5122
High Precision (90%)	76.8	272
Quad ($5N$) + HiPrec (90%)	76.8	263
$N\log^2 N$ ($3\log N$) + HP (90)	76.8	266
Linear (16) + HiPrec (90%)	76.8	264

Table 3: Speed and accuracy of exact-inference CYK parser on WSJ section 23 under various constraint conditions, including combining various complexity bound constraints and high precision constraints.

limitations prevent us from including scatter plots similar to those in Figure 3 for the constraint combination trials, which show that the observed parsing time of shorter sentences is typically identical under each constraint set, while the parsing time of longer sentences tends to differ more under each condition and exhibit characteristics of the complexity bounds. Thus by combining high-precision and complexity constraints, we combine typical-case efficiency benefits with worst-case complexity bounds.

Note that these speedups are achieved with no additional techniques for speeding up search, i.e., modulo the cell closing mechanism, the CYK parsing is exhaustive—it explores all possible category combinations from the open cells. Techniques such as coarse-to-fine or A* parsing, the use of an agenda, or setting of probability thresholds on entries in cells—these are all orthogonal to the current approach, and could be applied together with them to achieve additional speedups. However, none of these other techniques provide what the current methods do: a complexity bound that will hold even in the worst case.

To validate the selected operating points on a different section, Table 3 presents speed and accuracy results on the test set (WSJ section 23) for the exact-inference CYK parser.

We also conducted similar preliminary trials for parsing the Penn Chinese Treebank (Xue et al., 2004), which contains longer sentences and different branching characteristics in the induced grammar. Results are similar to those shown here, with chart constraints providing both efficiency and accuracy gains.

7 Conclusion

We have presented a method for constraining a context-free parsing pipeline that provably achieves linear worst case complexity. Our method achieves comparable observed performance to the quadratic complexity method previously published in Roark and Hollingshead (2008). We were motivated to pursue this method by the observed linear parsing time achieved with the quadratic bound constraints, which suggested that a tighter complexity bound could be achieved without hurting performance.

We have also shown that combining methods for achieving complexity bounds—which are of primary utility for longer strings—with methods for achieving strong observed typical case speedups can be profitable, even for shorter strings. The resulting combination achieves both typical speedups and worst-case bounds on processing.

The presented methods may not be the only way to achieve these bounds using tagger pre-processing of this sort, though they do have the virtue of very simple constraint selection. More complicated methods that track, in fine detail, how many cells are open versus closed, run the risk of a constraint selection process that is itself quadratic in the length of the string, given that there are a quadratic number of chart cells. Even so, the presented methods critically control midpoints for all cells only via the set E (words that can end a multi-word constituent) and ignore B. More complicated methods for using both sets that also achieve linear complexity (perhaps with a smaller constant), or that achieve $O(N\log N)$ complexity rather than $O(N\log^2 N)$, may exist.

Acknowledgments

This research was supported in part by NSF Grant #IIS-0447214 and DARPA grant #HR0011-08-1-0016. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF or DARPA.

References

D. Blaheta and E. Charniak. 1999. Automatic compensation for parser figure-of-merit flaws. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics (ACL)*, pages 513–518.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics*, pages 132–139.

J. Cocke and J.T. Schwartz. 1970. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, NYU.

B. Djordjevic, J.R. Curran, and S. Clark. 2007. Improving the efficiency of a wide-coverage CCG parser. In *Proceedings of the 10th International Workshop on Parsing Technologies (IWPT)*, pages 39–47.

E. Glaysher and D. Moldovan. 2006. Speeding up full syntactic parsing by leveraging partial parsing decisions. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 295–300.

K. Hall and M. Johnson. 2004. Attention shifting for parsing speech. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 40–46.

K. Hollingshead and B. Roark. 2007. Pipeline iteration. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 952–959.

K. Hollingshead, S. Fisher, and B. Roark. 2005. Comparing and combining finite-state and context-free parsers. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 787–794.

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.

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.

A. Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.

B. Roark and K. Hollingshead. 2008. Classifying chart cells for quadratic complexity context-free inference. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 745–752.

N. Xue, F. Xia, F. Chiou, and M. Palmer. 2004. The Penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 10(4):1–30.

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

Improved Syntactic Models for Parsing Speech with Repairs*

Tim Miller
Department of Computer Science and Engineering
University of Minnesota, Twin Cities
tmill@cs.umn.edu

Abstract

This paper introduces three new syntactic models for representing speech with repairs. These models are developed to test the intuition that the erroneous parts of speech repairs (reparanda) are not generated or recognized as such while occurring, but only after they have been corrected. Thus, they are designed to minimize the differences in grammar rule applications between fluent and disfluent speech containing similar structure. The three models considered in this paper are also designed to isolate the mechanism of impact, by systematically exploring different variables.

1 Introduction

Recent work in recognition of speech with repairs has shown that syntactic cues to speech repair can improve both overall parsing accuracy and detection of repaired sections (Hale et al., 2006; Miller and Schuler, 2008; Johnson and Charniak, 2004). These techniques work by explictly modeling the structure of speech repair, specifically the tendency of repairs to follow unfinished constituents of the same category. This is the essence of what was termed the *well-formedness rule* by Willem Levelt (1983) in his psycholinguistic studies of repair.

The work presented here uses the same motivations as those cited above (to be described in more detail below), in that it attempts to model the syntactic structure relating unfinished erroneous constituents to the repair of those constituents. However, this work attempts to improve on those models by focusing on the generative process used by a speaker in creating the repair. This is done first by eschewing any labels representing the presence of an erroneous constituent while processing the text. This modeling representation reflects the intuition that speakers do not intend to generate erroneous speech – they intend their speech to be fluent, or a correction to an error, and can stop very quickly when an error is noticed. This corresponds to Levelt's *Main Interruption Rule*, which states that a speaker will "Stop the flow of speech immediately upon detecting the occasion of repair." Rather than attempting to recognize a special syntactic category called EDITED during the processing phase, this work introduces the REPAIRED category to signal the *ending* of a repaired section only.

The second part of the modeling framework is the use of a *right-corner transform* on training data, which converts phrase-structure trees into heavily left-branching structures. This transformation has been shown to represent the structure of unfinished constituents like those seen in speech repair in a natural way, leading to improved detection of speech repair (Miller and Schuler, 2008).

Combining these two modeling techniques in a bottom-up parsing framework results in a parsing architecture that is a reasonable approximation to the sequential processing that must be done by the human speech processor when recognizing spoken language with repairs. This parser also recognizes sentences containing speech repair with better accuracy than the previous models on which it is based.

*This research was supported by NSF CAREER award 0447685. The views expressed are not necessarily endorsed by the sponsors.

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 656–664,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

Therefore, these syntactic models hold promise for integration into systems for processing of streaming speech.

1.1 Speech Repair Terminology

A speech repair occurs when a speaker decides to interrupt the flow of speech and restart part or all of an utterance. Typically speech repair structure (Shriberg, 1994) is considered to contain a *reparandum*, or the part of the utterance to be replaced, and an *alteration*, which is meant to replace the reparandum section. There are also frequently *editing terms* (for example, 'uh' and 'um') between the reparandum and alteration, which may be used to signal the repair, or to indicate that the speaker is thinking, or just to maintain control of the dialogue.

1.2 Related Work

This work is related to that of Hale et al.(2006) in that it attempts to model the syntactic structure of speech repair. In that paper speech repair detection accuracy was increased by explicitly accounting for the relation between reparanda category and alteration category. This was done by so-called "daughter annotation," which expanded the set of EDITED categories by appending the category below the EDITED label to the end of the EDITED label – for example, a noun phrase (NP) reparanda would be of type EDITED-NP. In addition, this approach made edit detection easier by propagating the -UNF label attached to the rightmost unfinished constituent up to the EDITED label. These two changes in combination allow the parser to better recognize when a reparandum has occurred, and to make siblings of reparanda and alterations with the same basic category label.

Another model of speech repair that explicitly models the structure of speech repair is that of Johnson and Charniak (2004). That model has a different approach than the context-free parsing approach done in the present work. Instead, they run a tree-adjoining grammar (TAG) parser which traces the overlapping words and part-of-speech tags that occur in the reparandum and alteration of a speech repair. This approach is highly accurate at detecting speech repairs, and allows for downstream processing of cleaned up text to be largely free of speech repair, but due to its TAG component it may present difficulties incorporating into an architecture that operates on streaming text or speech.

This work is also similar in aim to a component of the parsing and language modeling work of Roark and Johnson (1999), which used right-binarization in order to delay decision-making about constituents as much as possible. For example, the rule

$$NP \rightarrow DT \;\; NN$$

might be right-binarized as two rules:

$$NP \rightarrow DT \;\; NP\text{-}DT$$

and

$$NP\text{-}DT \rightarrow NN$$

The result of this binarization is that when predicting the noun phrase (NP) rule, a top-down parser is delaying making any commitments about the category following the determiner (DT). This delay in prediction means that the parser does not need to make any predictions about whether the next word will be, e.g., a common noun (NN), plural noun (NNS), or proper noun (NNP), until it sees the actual next word.

Similarly, the model presented in this work aims to delay the decision to create a speech repair as much as possible. This is done here by eliminating the EDITED category (representing a reparandum) during processing, replacing it with a REPAIRED category which represents the alteration of a speech repair, and by eliminating implicit cues about repair happening before a decision to repair should be necessary.

Finally, this work is most directly related to that of Miller and Schuler (2008). In that work, the authors used a right-corner transform to turn standard phrase-structure trees into highly left-branching trees with sub-tree category labels representing incomplete but in-progress constituent structure. That structure was shown to have desirable properties in the representation of repair in syntax trees, and this work leverages that insight, while attempting to improve the input representation such that the right-corner representation does not require the parser to make any assumptions or decisions earlier than necessary.

2 Syntactic Model

This section will first describe the default representation scheme for speech repair in the Switchboard corpus and the standard representation after application of a right-corner transform, and then describe why there are shortcomings in both of these representations. Descriptions of several alternative models follow, with an explanation of how each of them is meant to address the shortcomings seen in previous representations. These models are then evaluated in Section 3.

2.1 Standard Repair Annotation

The standard representation of speech repair in the Switchboard corpus makes use of one new category label (EDITED), to represent a reparandum, and a new dash-tag (-UNF), representing the lowest unfinished constituent in a phrase. An example tree with both EDITED and -UNF tags is shown in Figure 1.

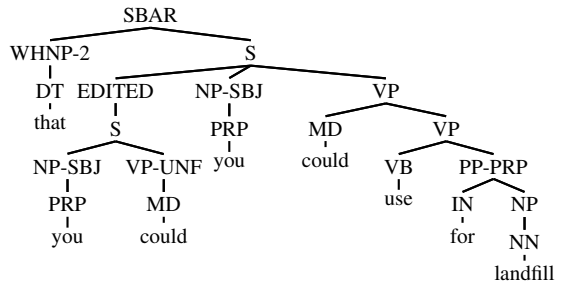


Figure 1: A fragment of a standard phrase-structure tree from the development set, containing both an EDITED constituent and an -UNF tag.

This sentence contains a restarted sentence (S) constituent, in which the speaker started by saying "you could", then decided to restart the phrase, in this case without changing the first two words. One important thing to notice is that the EDITED label contains no information about the structure beneath it. As a result, a parser trained on this default annotation has no information about the attempted constituent type, which, in the case of restarts would obviously be beneficial. As described above, the work by Hale et al. using daughter annotation was meant to overcome this shortcoming.

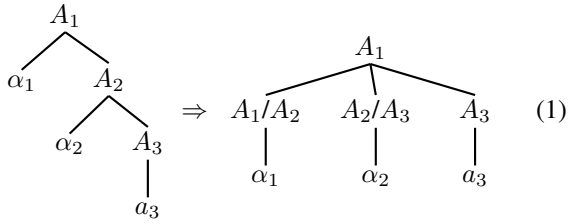
Another shortcoming of this annotation scheme to consider is that the EDITED tag is not meaningful with respect to constituent structure. Attempting to learn from this structure, for example a probabilistic context-free grammar, will result in the rule that a sentence (S) consists of a reparandum, a noun phrase, and a verb phrase, which is an odd way of thinking about both constituent structure and meaning. A more intuitive understanding might be that a sentence may consist of a noun phrase followed by a verb phrase, and during the production of that rule, an interruption may occur which causes the rule to restart.

2.2 Right-Corner Transform

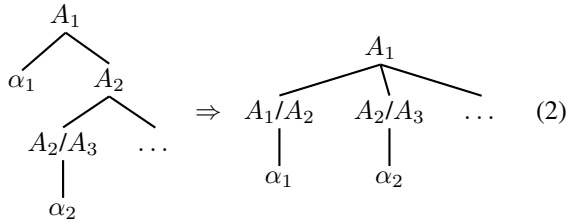
The work described above by Miller and Schuler (2008) uses a right-corner transform. This transform turns right-branching structure into left-branching structure, using category labels that use a "slash" notation α/γ to represent an incomplete constituent of type α "looking for" a constituent of type γ in order to complete itself. Figure 2 shows the right-corner transformed tree from above.

This transform first requires that trees be binarized. This binarization is done in a similar way to Johnson (1998) and Klein and Manning (2003).

Rewrite rules for the right-corner transform are as follows, first flattening right-branching structure:[1]



(1)



(2)

then replacing it with left-branching structure:

[1]Here, all A_i denote nonterminal symbols, and α_i denote subtrees ; the notation $A_1{:}\alpha_0$ indicates a subtree α_0 with label A_1; and all rewrites are applied recursively, from leaves to root. In trees containing repairs, the symbol ET represents any number of editing terms and the sub-structure within them.

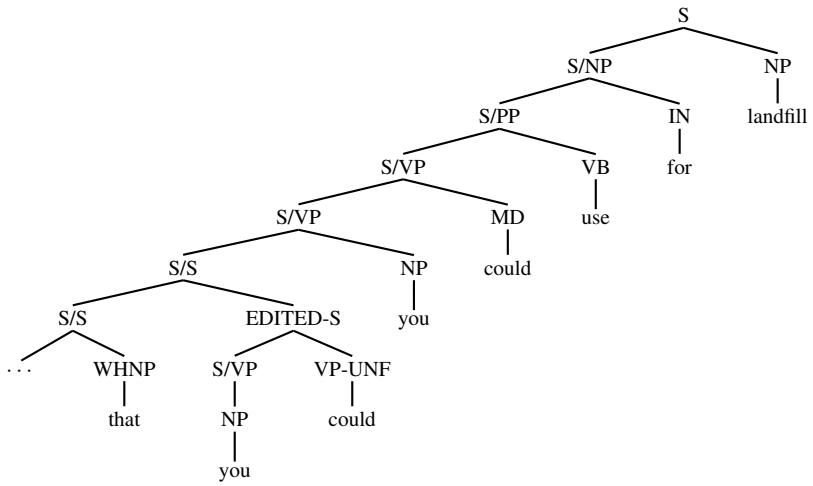
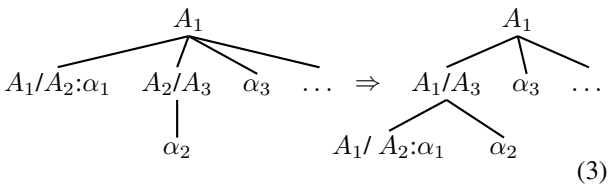


Figure 2: Right-corner transformed tree fragment.



(3)

This representation has interesting properties, which work well for speech repair. First, the left-branching structure of a repair results in reparanda that only require one special repair rule application, at the last word in the reparandum. Second, the explicit representation of incomplete constituents allows many reparanda to seamlessly integrate with the rest of the parse tree, with the EDITED label essentially acting as an instruction to the parser to maintain the current position in the unfinished constituent. This subtle second point is illustrated in the tree in Figure 2. After the EDITED section is detected, it combines with a category label S/S to form another sub-tree with category label S/S, essentially acting as a null op in a state machine looking to complete a phrase of type S.

This representation also contains problems, however. First, note that the (bottom-up) parser uses one set of rules to combine the reparandum with the current state of the recognition, and another set of rules when combining the alteration with the previous input. While it is a benefit of this approach that both rule sets are made up of fluent speech rules, their way of combining nonetheless requires an early premonition of the repair to occur. If anything, the repair should require special rule applications, but in this representation it is still the case that the reparandum looks different and the alteration looks "normal."

A better model of repair from a recognition perspective would recognize the reparandum as fluent, since they are recognized as such in real time, and then, when noticing the repeated words, declare these new words to be a repair section, and retroactively declare the original start of the phrase to be a reparandum. It is this conception of a recognition model that forms part of the basis for a new syntactic model of speech repair in Section 2.3.

A second problem with this representation is evident in certain multi-word repairs such as the one in Figure 2 that require an extra right branch off of the main left branching structure of the tree. As a result, a multi-word reparandum structure requires an extra unary rule application at the left-corner of the sub-tree, in this case S/VP, relative to the inline structure of the fluent version of that phrase. This extra rule will often be nearly deterministic, but in some cases it may not be, which would result essentially in a penalty for starting speech repairs. This may act to discourage short repairs and incentivize longer reparanda, across which the penalty would be amortized. This incentive is exactly backwards, since reparanda tend to be quite short.

The next section will show how the two issues mentioned above can be resolved by making mod-

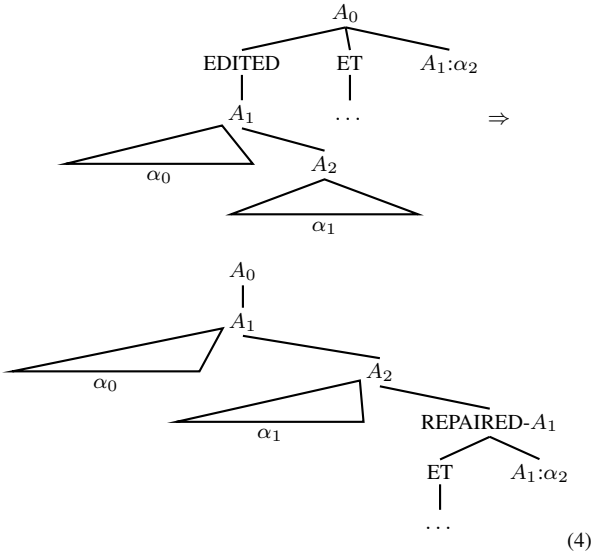
ifications to the original structure of trees containing repairs.

2.3 Modified Repair Annotation

The main model introduced in this paper works by turning the original repair into a right-branching structure as much as possible. As a result, the right-corner transformed representation has very flat structure, and, unlike the standard right-corner transformed representation described above, does not require a second level of depth in the tree with different rule applications. This can also be an important consideration for speech, since there are parsers that can operate in asymptotically linear time by using bounded stacks, and flat tree structure minimizes the amount of stack space required.

This model works by using an "interruption" model for the way a repair begins. The interruption model works on restarted constituents, by moving the repaired constituent (the alteration) to be the right-most child of the original EDITED constituent. The EDITED label is then removed, and a new REPAIRED label is added. This of course makes the detection of EDITED sections possible only retrospectively, by noting a REPAIRED section of a certain syntactic category, and tracing back in the tree to find the closest ancestor of the same category.

This can be illustrated schematically by the following rewrite rule:



(4)

Figure 3 shows how the example tree from Figure 1 looks when transformed in this manner. The result of these transformations may appear odd, but it is important to note that it is merely an intermediate stage between the "standard" representation with an EDITED label, representing the post-recognition understanding of the sentence, and the right-corner representation in which recognition actually occurs. This right-corner representation can be seen in Figure 2.3.

This representation is notable in that it looks exactly the same after the first word of the repair ('you') as the later incarnation of the same word in the alteration. After the second word ('could'), the repair is initiated, and here a repair rule is initiated. It should be noted, however, that strictly speaking the only reason the REPAIRED category needs to exist is to keep track of edits for the purpose of evaluating the parser. It serves only a processing purpose, telling the parser to reset what it is looking for in the incoming word stream.

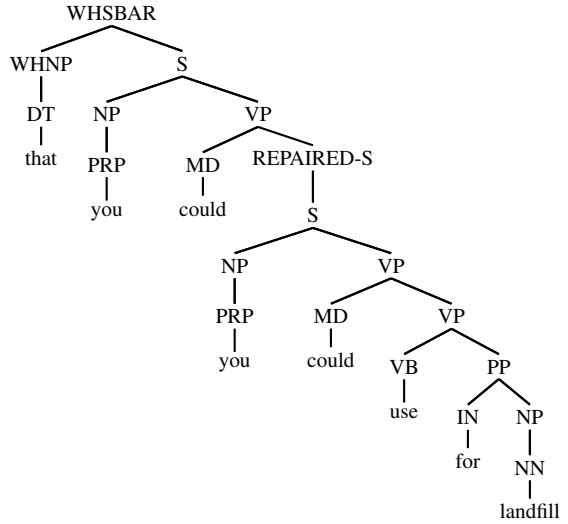


Figure 3: REPAIRED-INT transformation

The next model attempts to examine the impact of two different factors in the REPAIRED-INT representation above. That representation had the side effect of creating special rules off of the alteration (REPAIRED) node, and it is difficult to assign praise or blame to the performance results of that model without distinguishing the main modification from the side effects. This can be rectified by proposing another model that similarly eliminates the EDITED label for reparanda, and uses a new label REPAIRED for the alteration, but that

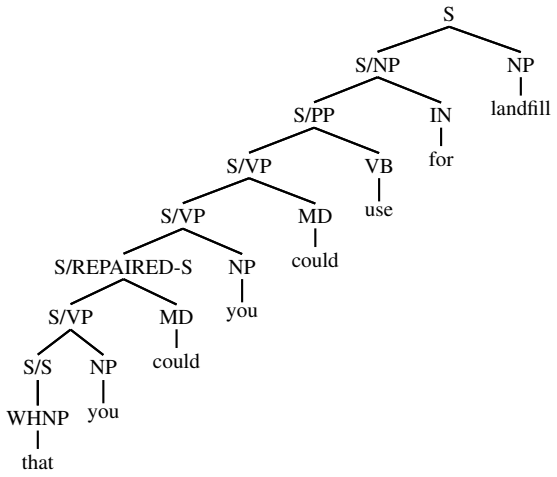


Figure 4: REPAIRED-INT + right-corner transformation

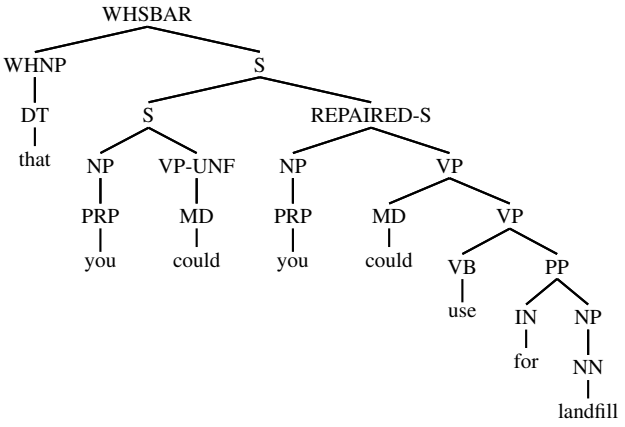
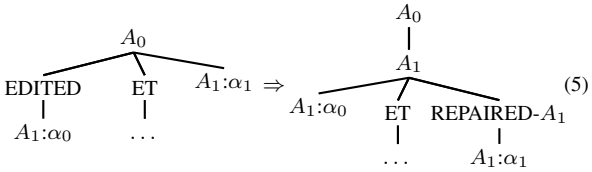


Figure 5: REPAIRED-BIN transformation

does not satisfy the desire to have reparanda occur inline using the "normal" rule combinations. This model does, however, still have special rules that the REPAIRED label will generate. Thus, if this model performs equally well (or equally as poorly) as REPAIRED-INT, then it is likely due to the model picking up strong signals about an alteration rule set. This modification involves rewriting the original phrase structure tree as follows:



(5)

A tree with this annotation scheme can be seen in Figure 5, and its right-corner counterpart is shown in Figure 6.

The final modification to examine acts effectively as another control to the previous two annotation schemes. The two modifications above are essentially performing two operations, first acting to binarize speech repairs by lumping a category of type X with a category of type EDITED-X, and then explicitly marking the repair but not the reparandum. This modification tests whether simply adding an extra layer of structure can improve performance while retaining the standard speech repair annotation including the EDITED category label. This modification will be denoted EDITED-BIN.

EDITED-BIN trees are created using the following rewrite rule:

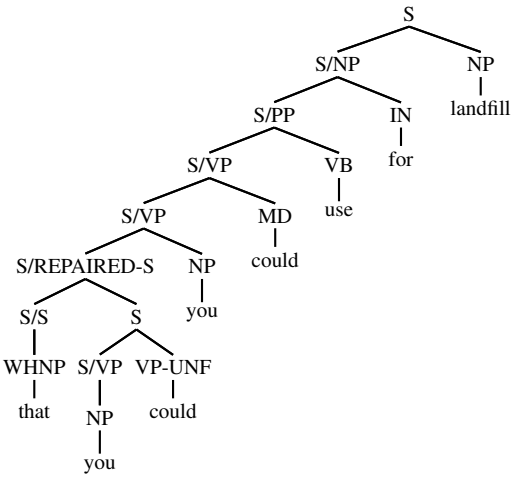
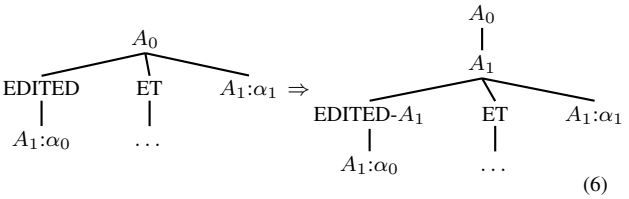


Figure 6: REPAIRED-BIN + right-corner transformation



(6)

After this transform, the tree would look identical to the REPAIRED-BIN tree in Figure 5, except the node labeled 'REPAIRED-S' is labeled 'S', and its left sibling is labeled 'EDITED-S' instead of 'S.' An EDITED-BIN tree after right-corner transformations is shown in Figure 7. This explicit binarization of speech repairs may be effective in its own right, because without it, a 'brute force' binarization must be done to format the tree before applying the right-corner transform, and that process in-

volves joining chains of categories with underscores into right-branching super-categories. This process can result in reparanda categories in unpredictable places in the middle of lengthy super-categories, making data sparse and less reliable.

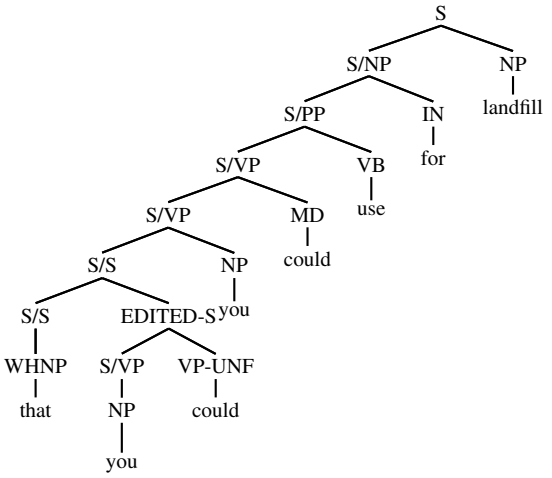


Figure 7: EDITED-BIN + right-corner transformation

3 Evaluation

The evaluation of this model was performed using a probabilistic CYK parser[2]. This parser operates in a bottom-up fashion, building up constituent structure from the words it is given as input. This parsing architecture is a good match for the structure generated by the right-corner transform because it does not need to consider any categories related to speech repair until the repaired section has been completed. Moreover, the structure of the trees means that the parser is also building up structure from left to right. That mode of operation is useful for any model which purports to be potentially extensible to speech recognition or to model the human speech processor. In contrast, top-down parsers require exhaustive searches, meaning that they need to explore interpretations containing disfluency, even in the absence of syntactic cues for its existence.

These experiments used the Switchboard corpus (Godfrey et al., 1992), a syntactically-annotated corpus of spontaneous dialogues between human interlocutors. This corpus is annotated for phrase structure in much the same way as the Penn Treebank Wall Street Journal corpus, with the addition of several speech-specific categories as described in Section 2.1. For training, trees in sections 2 and 3 of this corpus were transformed as described in Section 2, and rule probabilities were estimated in the usual way. For testing, trees in section 4, subsections 0 and 1, were used. Data from the tail end of section 4 (subsections 3 and 4) was used during development of this work.

Before doing any training or testing, all trees in the data set were stripped of punctuation, empty categories, typos, all categories representing repair structure, and partial words – anything that would be difficult or impossible to obtain reliably with a speech recognizer. A baseline parser was then trained and tested using the split described above, achieving standard results as seen in the table below. For a fair comparison to the evaluation in Hale et al. (2006), the parser was given part-of-speech tags along with each word as input. The structure obtained by the parser was then in the right-corner format. For standardized scoring, the right-corner transform, binarization, and augmented repair annotation were undone, so that comparison was done against the nearly pristine test corpus. Several test configurations were then evaluated, and compared to three baseline approaches.

The two metrics used here are the standard Parseval F-measure, and Edit-finding F. The first takes the F-score of labeled precision and recall of the nonterminals in a hypothesized tree relative to the gold standard tree. The second measure marks words in the gold standard as edited if they are dominated by a node labeled EDITED, and measures the F-score of the hypothesized edited words relative to the gold standard (recall in this case is percentage of actual edited words that were hypothesized as edited, and precision is percentage of hypothesized edited words that were actually edited).

The first three lines in the table refer to baseline approaches to compare against. "Plain" refers to a configuration with no modifications other than the removal of repair cues. The next result shown is a reproducton of the results from Hale et al. (2006) (described in section 1.2)[3]. The next line ("Standard

[2]The specific parser used is the Stanford parser described in Klein and Manning(2003), but run in "vanilla PCFG" mode.

[3]The present work compares to the standard CYK parsing result from that paper, and not the result from a heavily optimized parser using lexicalization.

Right Corner") is a reproduction of the results from Miller and Schuler (2008).

The following three lines contain the three experimental configurations. First, the configuration denoted EDITED-BIN refers to the simple binarized speech repair described in Section 2.3 (Equation 6). REPAIRED-BIN refers to the binarized speech repair in which the labels are basically reversed from EDITED-BIN (Equation 5). Finally, REPAIRED-INT refers to the speech repair type where the REPAIRED category may be a child of a non-identity category, representing an interruption of the outermost desired constituent (Equation 4).

System Configuration	*Parseval-F*	*Edited-F*
Baseline	71.03	17.9
Hale et al.	$68.47^{\dagger\dagger}$	$37.9^{\dagger\dagger}$
Standard Right Corner	$71.21^{\dagger\dagger}$	$30.6^{\dagger\dagger}$
EDITED-BIN	$69.77^{**\ \dagger\dagger}$	$38.9^{**\ \dagger\dagger}$
REPAIRED-BIN	71.37^{*}	$31.6^{**\ \dagger\dagger}$
REPAIRED-INT	71.77^{**}	$39.2^{**\ \dagger\dagger}$

Table 1: Table of parsing results. Star (*) indicates significance relative to the 'Standard Right Corner' baseline ($p < 0.05$), dagger (†) indicates significance relative to the 'Baseline' labeled result ($p < 0.05$). Double star and dagger indicate highly significant results ($p < 0.001$).

Significance results were obtained by performing a two-tailed paired Student's t-test on both the Parseval-F and Edit-F per-sentence results. This methodology is not perfect, since it fails to account for the ease of recognition of very short sentences (which are common in a speech corpus like Switchboard), and thus slightly underweights performance on longer sentences. This is also the explanation for the odd effect where the 'REPAIRED-BIN' and 'REPAIRED-INT' results achieve significance over the 'Standard Right Corner' result, but not over the 'Baseline' result. However, the simplest alternative – weighting each sentence by its length – is probably worse, since it makes the distributions being compared in the t-test broadly distributed collections of unlike objects, and thus hard to interpret meaningfully.

These results show a statistically significant improvement over previous work in overall parsing accuracy, and obvious (as well as statistically significant) gains in accuracy recognizing edited words (reparanda) with a parser. The REPAIRED-INT approach, which makes repair structure even more highly left-branching than the standard right-corner transform, proved to be the most accurate approach. The superior performance according to the EDIT-F metric by REPAIRED-INT over REPAIRED-BIN suggests that the improvement of REPAIRED-INT over a baseline is not due simply to a new category. The EDITED-BIN approach, while lowering overall accuracy slightly, does almost as well on EDITED-F as REPAIRED-INT, despite having a very different representation of repair. This suggests that there are elements of repair that this modification recognizes that the others do not. This possibility will be explored in future work.

Another note of interest regards the recovery of reparanda in the REPAIRED-INT case. As mentioned in Section 2.3, the EDITED section can be found by tracing upwards in the tree from a REPAIRED node of a certain type, to find an non-repaired ancestor of the same type. This makes an assumption that repairs are always maximally local, which probably does not hurt accuracy, since most repairs actually are quite short. However, this assumption is obviously not true in the general case, since in Figure 3 for example, the repair could trace all the way back to the S label at the root of the tree in the case of a restarted sentence. It is even possible that this implicit incentive to short repairs is responsible for some of the accuracy gains by discounting long repairs. In any case, future work will attempt to maintain the motivation behind the REPAIRED-INT modification while relaxing hard assumptions about repair distance.

4 Conclusion

This paper introduced three potential syntactic representations for speech with repairs, based on the idea that errors are not recognized as such until a correction is begun. The main result is a new representation, REPAIRED-INT, which, when transformed via the right-corner transform, makes a very attractive model for speech with repairs. This representation leads to a parser that improves on other parsing approaches in both overall parsing accuracy and accuracy recognizing words that have been edited.

References

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proc. ICASSP*, pages 517–520.

John Hale, Izhak Shafran, Lisa Yung, Bonnie Dorr, Mary Harper, Anna Krasnyanskaya, Matthew Lease, Yang Liu, Brian Roark, Matthew Snover, and Robin Stewart. 2006. PCFGs with syntactic and prosodic indicators of speech repairs. In *Proceedings of the 45th Annual Conference of the Association for Computational Linguistics (COLING-ACL)*.

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 (ACL '04)*, pages 33–39, Barcelona, Spain.

Mark Johnson. 1998. PCFG models of linguistic tree representation. *Computational Linguistics*, 24:613–632.

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.

Willem J.M. Levelt. 1983. Monitoring and self-repair in speech. *Cognition*, 14:41–104.

Tim Miller and William Schuler. 2008. A unified syntactic model for parsing fluent and disfluent speech. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL '08)*.

Brian Roark and Mark Johnson. 1999. Efficient probabilistic top-down and left-corner parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 99)*.

Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis, University of California at Berkeley.

A model of local coherence effects in human sentence processing as consequences of updates from bottom-up prior to posterior beliefs

Klinton Bicknell and **Roger Levy**
Department of Linguistics
University of California, San Diego
9500 Gilman Dr, La Jolla, CA 92093-0108
{kbicknell,rlevy}@ling.ucsd.edu

Abstract

Human sentence processing involves integrating probabilistic knowledge from a variety of sources in order to incrementally determine the hierarchical structure for the serial input stream. While a large number of sentence processing effects have been explained in terms of comprehenders' rational use of probabilistic information, effects of local coherences have not. We present here a new model of local coherences, viewing them as resulting from a belief-update process, and show that the relevant probabilities in our model are calculable from a probabilistic Earley parser. Finally, we demonstrate empirically that an implemented version of the model makes the correct predictions for the materials from the original experiment demonstrating local coherence effects.

1 Introduction

The task of human sentence processing, recovering a hierarchical structure from a serial input fraught with local ambiguities, is a complex and difficult problem. There is ample evidence that comprehenders understand sentences incrementally, constructing interpretations of partial structure and expectations for future input (Tanenhaus et al., 1995; Altmann and Kamide, 1999). Many of the main behavioral findings in the study of human sentence processing have now been explained computationally. Using probabilistic models trained on large-scale corpora, effects such as global and incremental disambiguation preferences have been shown to be a result of the rational use of syntactic probabilities (Jurafsky, 1996; Hale, 2001; Narayanan and Jurafsky, 2001; Levy, 2008b; Levy et al., 2009). Similarly, a number of other effects in both comprehension and production have been modeled as resulting from rational strategies of languages users that take into account all the probabilistic information present in the linguistic signal (Genzel and Charniak, 2002; Genzel and Charniak, 2003; Keller, 2004; Levy and Jaeger, 2007).

One class of results from the literature that has not yet been explained in terms of a rational comprehender strategy is that of local coherence effects (Tabor et al., 2004; Gibson, 2006; Konieczny and Müller, 2007), cases in which it appears that the parser is systematically ignoring contextual information about possible syntactic structures and pursuing analyses that are probable only locally. These effects are problematic for rational models, because of the apparent failure to use all of the available information. This paper describes a new model of local coherence effects under rational syntactic comprehension, which proposes that they arise as a result of updating prior beliefs about the structures that a given string of words is likely to have to posterior beliefs about the likelihoods of those structures in context. The critical intuition embodied in the model is that larger updates in probability distributions should be more processing-intensive; hence, the farther the posterior is from the prior, the more radical the update required and the greater the processing load. Section 2 describes the problem of local coherences in detail and Section 3 describes existing models of the phenomenon. Following that, Sections 4–5 describe our model and its computa-

Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, pages 665–673,
Boulder, Colorado, June 2009. ©2009 Association for Computational Linguistics

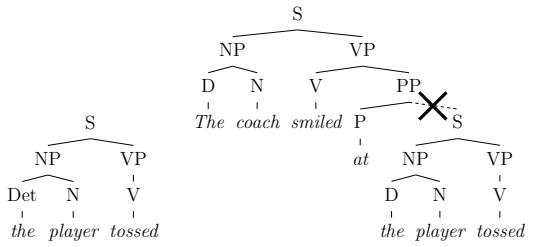


Figure 1: The difficulty of explaining local-coherence effects as traditional garden-pathing.

tion from a probabilistic Earley parser. Section 6 presents the results of an experiment showing that our model makes the correct predictions for the local coherence effects seen in the original paper by Tabor et al. (2004). Finally, Section 7 concludes and discusses the insight our model gives into human performance.

2 Local coherences

The first studies to report effects of local coherences are described in Tabor et al. (2004). In Experiment 1, they use a self-paced reading task and materials containing relative clauses (RCs) attached to nouns in non-subject position as in (1).

(1) a. The coach smiled at the player tossed a frisbee by the opposing team.
b. The coach smiled at the player who was tossed a frisbee by the opposing team.
c. The coach smiled at the player thrown a frisbee by the opposing team.
d. The coach smiled at the player who was thrown a frisbee by the opposing team.

Their experimental design crossed RC reduction with verb ambiguity. RCs are either reduced (1a,1c) or unreduced (1b,1d), and the RC verb is either lexically ambiguous between a past tense active and a past participle (1a–1b), or is unambiguously a past participle (1c–1d).

Tabor et al. point out that in one of these four conditions (1a) there is a locally coherent string *the player tossed a frisbee*. Out of context (e.g., if it were starting a sentence) this string would have a likely parse in which *tossed* is a past tense active verb, *the player* is its agent, and *a frisbee* is its theme (Figure 1, left). The preceding context within the sentence, however, should rule out this interpretation because *the player* appears within a PP and hence should not be able to be the subject of a new sentence (Figure 1, right). That is, given the preceding context, *the player tossed a frisbee* must begin a reduced RC, such that there is no local ambiguity. Thus, if comprehenders are making full use of the linguistic context, (1a) should be no more difficult than the other examples, except insofar as ambiguous verbs are harder than unambiguous verbs, and reduced RCs are harder than unreduced RCs, predicting there would be only the two main effects of RC reduction and verb ambiguity on reading times for the *tossed a frisbee* region.

Tabor et al., however, predict an interaction such that (1a) will have added difficulty above and beyond these two effects, because of the interference from the locally coherent parse of *the player tossed a frisbee*. Concordant with their predictions, they find an interaction in the *tossed a frisbee* region, such that (1a) is super-additively difficult. Because this result requires that an impossible parse influences a word's difficulty, it is in direct opposition to the predictions of theories of processing difficulty in which the probability of a word given context is the primary source of parsing difficulty, and more generally appears to be in opposition to any rational theory, in which comprehenders are making use of all the information in the linguistic context.

3 Existing models

With the results showing local coherence effects in mind, we can ask the question of what sorts of theories do predict these effects. This section briefly describes two recent examples of such theories. The first involves dynamical systems models to explain the effects and the second uses a mathematical model of the combination of bottom-up and top-down probabilistic information.

In Tabor and Hutchins's (2004) SOPARSE (self-organized parse) model, reading a word activates a set of lexically anchored tree fragments. Through spreading activation between compatible fragments and inhibition between incompatible ones, these tree fragments then compete in a process which is sensitive only to the local environment, i.e., ignoring the global grammatical context. Eventually, the sys-

tem stabilizes to the correct parse, and reading times for each word are modeled as the time the system takes to stabilize after reading a word. Stabilization takes longer for locally coherent regions because the locally coherent parse will be created and compete with the globally grammatical parse.

There are, however, unresolved issues with this model. The model has a number of free parameters, relating to the equations used for the competition, the method by which links between fragments are formed, as well as the question of precisely what tree fragments a given word will activate. While Tabor and Hutchins (2004) work out these questions in detail for the types of sentences they model, it is unclear how the model could be scaled up to make predictions for arbitrary types of sentences. That is, there is no principled system for setting the three types of parameters mentioned, and thus no clear interpretation of their values. The model put forward in this paper is an attempt to remedy this situation.

A recent proposal by Gibson (2006) can also explain some of the local coherence results. Gibson's proposal is that part-of-speech ambiguities have a special status in parsing; in effect, lexical part-of-speech ambiguities can be thought of as one-word local coherences. In this model, a probability function $\tilde{P}$ is calculated over part-of-speech tags given a word. This probability for tag t_i and a word w, $\tilde{P}(t_i|w)$, is proportional to the context-independent probability of t_i given the word w, $P(t_i|w)$ – the *bottom-up* component – multiplied by a smoothed probability P_s of the tag given the context – the *top-down* component:

$$\tilde{P}(t_i|w) = \frac{P(t_i|w)P_s(t_i|\text{context})}{\sum_{t \in T} P(t|w)P_s(t|\text{context})} \qquad (1)$$

Difficulty is predicted to be high when the probability $\tilde{P}$ of the correct tag is low.

Because the top-down probabilities are smoothed to allow for all possible parts-of-speech, any word which is lexically ambiguous will be more difficult to process, regardless of whether it is ambiguous or not in its context. This can thus explain some of the difference between the ambiguous and unambiguous verbs in Tabor et al. (2004). It is not clear, however, under such a model why the super-additive interaction would obtain—that is, why (1a) should be so much harder than (1b) starting at the word *tossed*. In addition, Gibson's model is a bit underspecified: he does not discuss how the top-down probabilities are calculated, nor what the precise linking hypothesis is between the final $\tilde{P}$ and reading times. Finally, it is not at all clear why the top-down expectations should be smoothed, since the smoothing actually has negative consequences on the processor's performance.

4 Parsing as belief update

The basic intuition behind the model presented here is that incrementally processing a sentence can be conceptualized as a process of updating one's beliefs. Such an analogy has been used to motivate surprisal-based theories of sentence processing (Hale, 2001; Levy, 2008a), where beliefs about the structure of a sentence after seeing the first $i-1$ words in the sentence, which we denote as w_0^{i-1}, are updated upon encountering w_i. In this case, the *surprisal* of a word ($-\log P(w_i|w_0^{i-1})$) is equivalent to the Kullback-Leibler divergence of the beliefs after w_0^i from the beliefs after w_0^{i-1} (Levy, 2008a). Our model focuses on another belief-update process in sentence processing: updating beliefs about the structures that a string of words is likely to have independent of context to beliefs about what structures it is likely to have in context. A bit more formally, it views the process of integrating a string of words w_i^j into a sentence as beginning with a 'bottom-up' prior distribution of syntactic structures likely to span w_i^j and integrating that with 'top-down' knowledge from the previous words in the sentence w_0^i in order to reach a posterior distribution conditioning on w_0^j over which structures actually can span w_i^j. This belief update process can be viewed as a rational reconstruction of the Tabor and Hutchins (2004) model, where – instead of the system dynamics of competition between arbitrary tree fragments – differences between prior and posterior probability distributions over syntactic structures determine processing difficulty.

More formally still, when integrating w_i^j into a sentence, for each syntactic category X, we can define the prior probability conditioned only on w_i^j that w_i^j will form the beginning of that category, i.e., that an X exists which begins at index i and spans at least

through j:

$$\text{Prior:} \quad P(X_i^{k \geq j} | w_i^j) \tag{2}$$

It is important to note here that this prior probability is conditional only on the value of w_i^j and not the values of i or j; that is, in the prior probability, i and j should be interpreted merely as a way to coindex the start and end points of the string of words being integrated with a category X potentially spanning them, and not as making reference to position in the full sentence string.

For each category X, this prior probability will be updated to the posterior probability of that category spanning w_i^j given all the words seen so far:

$$\text{Posterior:} \quad P(X_i^{k \geq j} | w_0^j) \tag{3}$$

In the equation for the posterior, of course, the indices i and j are positions in the sentence string, and not merely coindices.

Given these prior and posterior beliefs, we predict difficulty to arise in cases where the prior requires substantial modification to reach the posterior, that is, cases in which the prior and posterior make substantially different predictions for categories. A strong local coherence will have sharply different prior and posterior distributions, causing difficulty. We represent the prior and posterior beliefs as vectors of the probabilities of each syntactic category spanning w_i^j, and measure M_{ij}, the amount of modification required, as the summed K-L divergence of the prior from the posterior vector. That is, if N is the set of nonterminals in the grammar, the size of the belief update is modeled as[1]

$$M_{ij} \stackrel{\text{def}}{=} \sum_{X \in N} D\left(P(X_i^{k \geq j} | w_0^j) \,||\, P(X_i^{k \geq j} | w_i^j)\right)$$

In the remainder of the paper, we show how to compute M_{ij} by using Bayesian inference on quantities calculated in ordinary probabilistic incremental Earley parsing with a stochastic context-free grammar (SCFG), and show that our model makes the correct predictions using an SCFG for English on the original local-coherences experiment of Tabor et al. (2004).

[1]Note that for each syntactic category $X \in N$, the probability distribution $P(X_i^{k \geq j} | I)$ for some information I is over a binary random variable indicating the presence of X. The different syntactic categories X that could span from i to any k are not mutually exclusive, hence we cannot define size of belief update as a single K-L divergence defined over multinomial distributions.

5 Computing priors and posteriors

For SCFGs, a probabilistic Earley parser (Earley, 1970; Stolcke, 1995) provides the basic quantities we need to compute the prior (2) and posterior (3) for each category X. Following Stolcke, we use capital Latin characters to denote non-terminal categories and use lowercase Greek characters to denote (possibly null) sequences of terminals and non-terminals. We write the probability that a non-terminal X can be recursively rewritten by SCFG rules as a certain series of symbols μ by

$$P(X \stackrel{*}{\Rightarrow} \mu)$$

An edge built from the rule $X \rightarrow \lambda\mu$ where λ has been recognized as beginning at position i and ending at position j is denoted

$$j : X_i \rightarrow \lambda . \mu$$

The forward probability of that edge at position j, α_j, is defined to be the joint probability that the root node will generate all words recognized so far w_0^j as well as the edge

$$\alpha_j(X_i \rightarrow \lambda . \mu)$$

With this terminology, we are now in a position to describe how we calculate the posterior and prior probability vectors for our model.

5.1 Calculating the posterior

To calculate the posterior, we first use the definition of conditional probability to rewrite it as

$$P(X_i^{k \geq j} | w_0^j) = \frac{P(X_i^{k \geq j}, w_0^j)}{P(w_0^j)}$$

In a context-free grammar, given the syntactic category that dominates a string of words, the words' probability is independent from everything outside the category. Thus, this is equivalent to

$$\begin{aligned} P(X_i^{k \geq j} | w_0^j) &= \frac{P(w_o^i, X_i) P(w_i^j | X_i^{k \geq j})}{P(w_0^j)} \\ &= \frac{P(S \stackrel{*}{\Rightarrow} w_0^i X \nu) P(X \stackrel{*}{\Rightarrow} w_i^j \mu)}{P(S \stackrel{*}{\Rightarrow} w_0^j \lambda)} \end{aligned}$$

5.1.1 Posterior: the numerator's first term

The first term in the numerator $P(S \stackrel{*}{\Rightarrow} w_0^i X\nu)$ can be computed from a parse of w_0^i by summing forward probabilities of the form

$$\alpha_i(X_i \rightarrow .\mu) \tag{4}$$

5.1.2 Posterior: the denominator

Similarly, the denominator $P(S \stackrel{*}{\Rightarrow} w_0^j \lambda)$ can be computed from a parse of w_0^j by summing forward probabilities of the form

$$\alpha_j(Y \rightarrow \lambda w_{j-1}^j .\mu) \tag{5}$$

for all Y. This is because the forward probability of a state is conditioned on generating all the previous words.

5.1.3 Posterior: the numerator's second term

The second term in the numerator $P(X \stackrel{*}{\Rightarrow} w_i^j \mu)$ for an arbitrary category X cannot necessarily be calculated from a probabilistic Earley parse of the sentence, because the parser does not construct states that are not potentially useful in forming a sentence (i.e., states that would have a forward probability of zero.) However, to calculate the probability of X generating words w_i^j we can parse w_i^j separately with a goal category of X. From this parse, we can extract the probability of w_i^j being generated from X in the same way as we extracted the probability of w_0^j being generated from S, i.e., as a sum of forward probabilities at j (Eq. 5).[2]

5.2 Calculating the prior

To calculate the prior, we first use Bayes rule to rewrite it as

$$P(X_i^{k \geq j} | w_i^j) = \frac{P(w_i^j | X_i^{k \geq j}) P(X_i^{k \geq j})}{P(w_i^j)} \tag{6}$$

Recall that at this point, i and j do not refer to index positions in the actual string but rather serve to identify the substring w_i^j of interest. That is, $P(w_i^j)$ denotes the probability that at an arbitrary point in an arbitrary sentence, the next $j - i$ words will be w_i^j, and $P(X_i^{k \geq j})$ denotes the probability that an arbitrary point in an arbitrary sentence will be the left edge of a category X that spans at least $j - i$ words. None of the three terms in Eq. 6 can be directly obtained. However, we can obtain a very good approximation of Eq. 6 as follows. First, we marginalize over the position within a sentence with which the left edge i might be identified:

$$P(X_i^{k \geq j} | w_i^j) = \sum_{i'=0,1,\ldots} \left(\frac{P(w_{i'}^j | X_{i'}^{k \geq j}) P(X_{i'}^{k \geq j})}{P(w_{i'}^j)} \right) P(i = i') \tag{7}$$

In Eq. 7, i' *is* identified with the actual string position within the sentence.

Second, we rewrite the first term in this sum with event space notation, using the event space given in Table 1.

Table 1: Event space for the prior

Event	Description	
E_0:	There are at least i' words	$\lvert w \rvert \geq i'$
E_1:	A category X begins at i'	$X_{i'}$
E_2:	An $X_{i'}$ spans at least through j	$X_{i'}^{k \geq j}$
E_3:	There are at least j words	$\lvert w \rvert \geq j$
E_4:	Words $w_{i'}^j$ are these specific $\tilde{w}_{i'}^j$	$w_{i'}^j = \tilde{w}_{i'}^j$

$$\frac{P(w_{i'}^j | X_{i'}^{k \geq j}) P(X_{i'}^{k \geq j})}{P(w_{i'}^j)} = \frac{P(E_{0,3,4} | E_{0\ldots3}) P(E_{0\ldots3})}{P(E_{0,3,4})} = \frac{P(E_4 | E_{0\ldots3}) P(E_{o\ldots3})}{P(E_{o,3,4})}$$

Applying the chain rule, we can further simplify.

$$= \frac{P(E_4 | E_{0\ldots3}) P(E_{1\ldots3} | E_0) P(E_0)}{P(E_{3,4} | E_0) P(E_0)} = \frac{P(E_4 | E_{0\ldots3}) P(E_{1\ldots3} | E_0)}{P(E_{3,4} | E_0)} = \frac{P(E_{2\ldots4} | E_0, E_1) P(E_1 | E_0)}{P(E_{3,4} | E_0)}$$

[2] To calculate the posterior, it is not necessary to parse w_i^j separately, since these states are only excluded from the parse when their forward probability is zero, in which case the first term in the numerator will also be zero. A separate parse is necessary, however, when using this term to calculate the prior.

Switching back from event space notation and substituting this term into Eq. 7, we now have

$$P(X_i^{k\geq j}|w_i^j) = \sum_{i'=0,1,\ldots} \left(\frac{P(w_{i'}^j|X_{i'},E_0)P(X_{i'}|E_0)}{P(w_{i'}^j|E_0)} \right) P(i=i') \quad (8)$$

Thus, by conditioning all terms on E_0, the presence of at least i' words, we have transformed the probabilities we need to calculate into these four terms, which are easier to calculate from the parser. We now consider how to calculate each of the terms.

5.2.1 Prior: the numerator's first term

The first term in the numerator can be simplified because our grammar is context-free:

$$\begin{aligned} P(w_{i'}^j|X_{i'},E_0) &= P(w_{i'}^j|X_{i'}) \\ &= P(X \stackrel{*}{\Rightarrow} w_{i'}^j) \end{aligned}$$

This can be computed as described in Section 5.1.3.

5.2.2 Prior: the numerator's second term

The second term in the numerator can be rewritten as follows:

$$\begin{aligned} P(X_{i'}|E_0) &= \frac{P(X_{i'},E_0)}{P(E_0)} \\ &= \frac{P(S \stackrel{*}{\Rightarrow} \mathring{w}_0^{i'} X \mu)}{P(S \stackrel{*}{\Rightarrow} \mathring{w}_0^{i'} \mu)} \end{aligned}$$

where $\mathring{w}_0^{i'}$ denotes any sequence of i' words. Given a value i' we can calculate both terms by parsing the string $\mathring{w}_0^i X$, where each word $\mathring{w}$ in $\mathring{w}_0^i X$ is a special word that can freely act as any preterminal. The denominator can then be calculated by summing the forward probabilities of the last word $\mathring{w}_{i-1}^i$ as in Eq. 5, and the numerator by summing the forward probability of X, as in Eq. 4.

5.2.3 Prior: the denominator

The denominator in the calculation of the prior can be calculated in a way analogous to the numerator's second term (Section 5.2.2):

$$\begin{aligned} P(w_{i'}^j|E_0) &= \frac{P(w_{i'}^j,E_0)}{P(E_0)} \\ &= \frac{P(S \stackrel{*}{\Rightarrow} \mathring{w}_0^{i'} w_{i'}^j \mu)}{P(S \stackrel{*}{\Rightarrow} \mathring{w}_0^{i'} \mu)} \end{aligned}$$

5.2.4 Prior: starting position probability

Finally, we must calculate the second term in Eq. 8, the probability of the starting position $P(i=i')$. Given that all our terms are conditional on the existence of all words in the sentence up to i' (E_0), the probability of a starting position $P(i)$ is the probability of drawing i' randomly from the set of positions in sentences generated by the grammar such that all words up to that position exist. For most language grammars, this distribution can be easily approximated by a sample of sentences generated from the SCFG, since most of the probability mass is concentrated in small indices.

6 Experiment

We tested the predictions of an implemented version of our model on the materials from Tabor et al. (2004). To generate quantitative predictions, we created a small grammar of relevant syntactic rules, and estimated the rule probabilities from syntactically annotated text. We calculated summed K-L divergence of the prior from the posterior vector for each word in the Tabor et al. items, and predict this sum to be largest at the critical region when the sentence has an effect of local coherence.

6.1 Methods

6.1.1 Grammar

We defined a small SCFG for the problem, and estimated its rule probabilities using the parsed Brown corpus. The resulting SCFG is identical to that used in Levy (2008b) and is given in Table 2.

6.1.2 Lexicon

Lexical rewrite probabilities for part-of-speech tags were also estimated using the entire parsed Brown corpus.

6.1.3 Materials

The materials were taken from Experiment 1 of Tabor et al. (2004). We removed 8 of their 20 items for which our trained model either did not know the critical verb or did not know the syntactic structure of some part of the sentence. For the other 12 items, we replaced unknown nouns (9 instances) and unknown non-critical verbs (2 instances), changed one plural noun to singular, and dropped one sentence-initial prepositional phrase.

Table 2: The SCFG used in Experiment 3. Rule weights given as negative log-probabilities in bits.

Rule		Weight
ROOT	→ S	0
S	→ S-base CC S-base	7.3
S	→ S-base	0.01
S-base	→ NP-base VP	0
NP	→ NP-base RC	4.1
NP	→ NP-base	0.5
NP	→ NP-base PP	2.0
NP-base	→ DT NN NN	4.7
NP-base	→ DT NN	1.9
NP-base	→ DT JJ NN	3.8
NP-base	→ PRP	1.0
NP-base	→ NNP	3.1
VP/NP	→ VBD NP	4.0
VP/NP	→ VBD	0.1
VP	→ VBD PP	2.0
VP	→ VBD NP	0.7
VP	→ VBD	2.9
RC	→ WP S/NP	0.5
RC	→ VP-pass/NP	2.0
RC	→ WP FinCop VP-pass/NP	4.9
PP	→ IN NP	0
S/NP	→ VP	0.7
S/NP	→ NP-base VP/NP	1.3
VP-pass/NP	→ VBN NP	2.2
VP-pass/NP	→ VBN	0.4

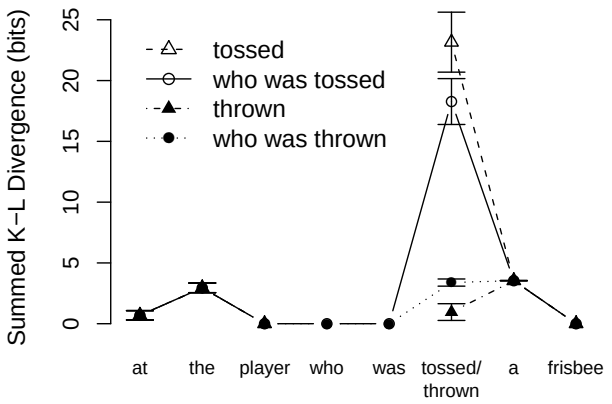


Figure 2: Summed K-L divergence of the prior from the posterior vectors at each word: Substring length 1

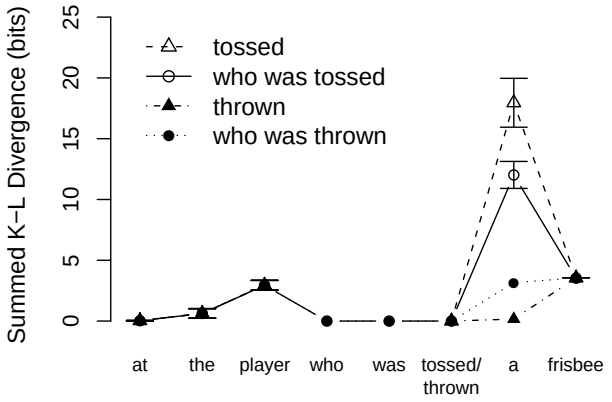


Figure 3: Summed K-L divergence of the prior from the posterior vectors at each word: Substring length 2

6.2 Procedure

For these 12 items, we ran our model on the four conditions in (1). For each word, we calculated the prior and posterior vectors for substrings of three lengths at w_i. The summed K-L divergence is reported for a substring length of 1 word using a prior of $P(X_{i-1}^{k\geq i}|w_{i-1}^{i})$, for a length of 2 using $P(X_{i-2}^{k\geq i}|w_{i-2}^{i})$, and for a length of 3 using $P(X_{i-3}^{k\geq i}|w_{i-3}^{i})$. For all lengths, we predict the summed divergence to be greater at critical words for the part-of-speech ambiguous conditions (1a,1b) than for unambiguous (1c,1d), because the part-of-speech unambiguous verbs cannot give rise to a prior that predicts for a sentence to begin. For a substring length of 3, we also predict that the divergence is superadditively greatest in the ambiguous reduced condition (1a), because of the possibility of starting a sentence with *the player tossed*.

6.3 Results

The results of the experiment are shown in Figures 2–4. For all three substring lengths, the model predicts difficulty to be greater in the ambiguous conditions at the critical words (*tossed/thrown a frisbee*). For 1-word substrings, the effect is localized on the critical verb (*tossed/thrown*), for 2-word substrings it is localized on the word directly following the critical verb (*tossed/thrown a*), and for 3-word substrings there are two effects: one on the critical verb (*the player tossed/thrown*) and one two words later (*tossed/thrown a frisbee*). Furthermore, for 3-word substrings, the effect is superadditively greatest for *the player tossed*. These results thus nicely confirm

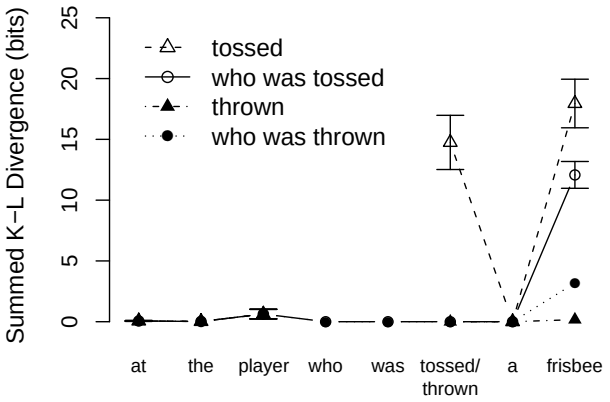


Figure 4: Summed K-L divergence of the prior from the posterior vectors at each word: Substring length 3

both of our predictions and demonstrate that a model in which large belief updates from a bottom-up prior to a posterior induce difficulty is capable of accounting for effects of local coherences.

7 Conclusion

This paper has described a model of local coherence effects in sentence processing, which views the process of integrating a string of words w_i^j into a sentence as a process of updating prior beliefs about the structures spanning those words to posterior beliefs. These prior beliefs are simply the probabilities of those structures given only the words being integrated, and the posterior beliefs are the probabilities given the entire sentence processed thus far. Difficulty is predicted to result whenever this update is large – which we model in terms of a large summed K-L divergence of the prior from the posterior vector. We demonstrated a method of normatively calculating these probabilities from probabilistic Earley parses and used this implemented model to make predictions for the materials for the original experimental result of effects of local coherences (Tabor et al., 2004). Our results demonstrated that the model predicts difficulty to occur at the correct part of the sentence in the correct condition.

We improve on existing models in two ways. First, we make predictions for where local coherences should obtain for an arbitrary SCFG, not just one particular class of sentences. This allows the model to scale up for use with a broad coverage grammar and to make predictions for arbitrary sentences, which was not possible with a model such as Tabor & Hutchins (2004).

Second, our model gives a rational basis to an effect which has typically been seen to result from irrationality of the human sentence processor. Specifically, the cost that our model describes of updating bottom-up prior beliefs to in-context posterior beliefs can be viewed as resulting from a rational process in the case that the bottom-up prior is available to the human sentence processor more rapidly than the in-context posterior. Interestingly, the fact that the prior is actually more difficult to compute than the posterior suggests that the only way it would be available more rapidly is if it is precomputed. Thus, our model provides the insight that, to the extent that comprehenders are behaving rationally in producing effects of local coherences, this may indicate that they have precomputed the likely syntactic structures of short sequences of words. While it may be unlikely that they calculate these probabilities for sequences directly from their grammar as we do in this paper, there could be a number of ways to approximate this prior: for example, given a large enough corpus, these probabilities could be approximated for any string of words that appears sufficiently often by merely tracking the structures the string has each time it occurs. Such a hypothesis for how comprehenders approximate the prior could be tested by manipulating the frequency of the relevant substrings in sentences with local coherences.

This work can be extended in a number of ways. As already mentioned, one logical step is using a broad-coverage grammar. Another possibility relates to the problem of correlations between the different components of the prior and posterior vectors. For example, in our small grammar, whenever a ROOT category begins, so does an S, an S-base, and an NP-base. Dimensionality reduction techniques on our vectors may be able to remove such correlations. These steps and more exhaustive evaluation of a variety of datasets remain for the future.

Acknowledgments

This research was supported by NIH Training Grant T32-DC000041 from the Center for Research in Language at UCSD to the first author.

References

Gerry T.M. Altmann and Yuki Kamide. 1999. Incremental interpretation at verbs: Restricting the domain of subsequent reference. *Cognition*, 73:247–264.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Dmitriy Genzel and Eugene Charniak. 2002. Entropy rate constancy in text. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 199–206, Philadelphia, July. Association for Computational Linguistics.

Dmitriy Genzel and Eugene Charniak. 2003. Variation of entropy and parse trees of sentences as a function of the sentence number. In Michael Collins and Mark Steedman, editors, *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 65–72, Sapporo, Japan. Association for Computational Linguistics.

Edward Gibson. 2006. The interaction of top-down and bottom-up statistics in the resolution of syntactic category ambiguity. *Journal of Memory and Language*, 54:363–388.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, volume 2, pages 159–166, New Brunswick, NJ. Associate for Computational Linguistics.

Daniel Jurafsky. 1996. A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, 20:137–194.

Frank Keller. 2004. The entropy rate principle as a predictor of processing effort: An evaluation against eye-tracking data. In Dekang Lin and Dekai Wu, editors, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 317–324, Barcelona, Spain, July. Association for Computational Linguistics.

Lars Konieczny and Daniel Müller. 2007. Local coherence interpretation in written and spoken language. Presented at the 20th Annual CUNY Conference on Human Sentence Processing. La Jolla, CA.

Roger Levy and T. Florian Jaeger. 2007. Speakers optimize information density through syntactic reduction. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 849–856, Cambridge, MA. MIT Press.

Roger Levy, Florencia Reali, and Thomas L. Griffiths. 2009. Modeling the effects of memory on human online sentence processing with particle filters. In *Proceedings of NIPS*.

Roger Levy. 2008a. Expectation-based syntactic comprehension. *Cognition*, 106:1126–1177.

Roger Levy. 2008b. A noisy-channel model of rational human sentence comprehension under uncertain input. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 234–243, Honolulu, Hawaii, October. Association for Computational Linguistics.

Srini Narayanan and Daniel Jurafsky. 2001. A Bayesian model predicts human parse preference and reading time in sentence processing. In T.G. Dietterich, S Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 59–65, Cambridge, MA. MIT Press.

Andreas Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

Whitney Tabor and Sean Hutchins. 2004. Evidence for self-organized sentence processing: Digging-in effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 30(2):431–450.

Whitney Tabor, Bruno Galantucci, and Daniel Richardson. 2004. Effects of merely local syntactic coherence on sentence processing. *Journal of Memory and Language*, 50:355–370.

Michael K Tanenhaus, Michael J Spivey-Knowlton, Kathleen M Eberhard, and Julie C Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268:1632–1634.

Author Index